高信頼化ソリューションガイド

― Lunux ―

2005年 12月 第1版

はじめに

PostgreSQL Plusは、PostgreSQLの機能を拡張し、Linuxプラットフォームで動作するオープン・ソースのデータベースシステムです。

本書は、PostgreSQL Plusの高信頼化ソリューションガイドです。

本書の目的

本書は、PostgreSQL Plusの持つ高信頼化機能を活用して、信頼性の高いシステムを構築する方法を説明しています。

本書の読者

本書は、PostgreSQL Plusの持つ高信頼化機能を活用して、信頼性の高いシステムを構築される方を対象としています。

なお、本書は、以下についての一般的な知識があることを前提に書かれています。

- PostgreSQLに関する知識
- PostgreSQL Plusに関する知識
- Linuxに関する知識

注意事項や参考情報

特に注意が必要な事項や、参考情報は、以下の形式で記述しています。

特に注意していただきたいことを記述しています。必ずお読みください。

知っていると役に立つ情報を記述しています。

その他の表記

項目	説明
“ ”	参照箇所、参照マニュアルを表しています。
【 】	画面、ボタン（オプションボタン）、チェックボックスなどは【】で囲んで記述してあります。以下に、例を示します。 例） 【セットアップ】画面 【次へ】ボタン 【リモートからのアクセスを許可する】チェックボックス

商標

Microsoft、Windowsは、米国 Microsoft Corporationの米国およびその他の国における登録商標または商標です。
NetscapeおよびNetscape Navigatorは、Netscape Communications Corporationの米国およびその他の国における商標です。
Linuxは、Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。
Red Hat、RPMおよびRed Hatをベースとしたすべての商標とロゴは、Red Hat, Inc.の米国およびその他の国における登録商標あるいは商標です。
TurboLinux は、ターボリナックス株式会社の登録商標です。
そのほか、本書に記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

なお、本マニュアルでは、(R)、TM表記を省略しています。

平成17年12月 初版

第1章　　高信頼性の強化

技術の進歩により、ハードウェアの信頼性は年々強化されています。しかし、デバイス異常によるデータの内容異常や、災害の発生によるデータベースのデータ、バックアップのデータ喪失などの不測事態を免れることはできません。データベースシステムの信頼性を高めるためには、これらの異常を迅速に検出し、異常の内容により適切に復旧できることが重要になってきます。ここでは、PostgreSQL Plusの機能を活用して、信頼性を向上させる以下のソリューションについて説明します。

- モニタリング
- ディザスタリカバリ

1.1 モニタリング

ここでは、PostgreSQL Plusのモニタリングの考え方について説明します。

モニタリングとは

データの異常状態を迅速に検出するためには、データの状態を定期的に監視するモニタリングが重要となります。
PostgreSQL Plusでは、以下に示すインスタンス資源の異常現象を検出した場合には、当該資源をアクセス禁止状態(閉塞状態ともいいます)にし、そのまま処理を続行することによってデータ破壊などの被害が広がることを防ぎます。

- データベースのディスク異常の検出
- ディスクの間欠障害によるデータ喪失の検出
- ハードウェアの不調によるデータ内容の異常の検出

さらに、適切な復旧のために、以下に示すインスタンス資源の異常発生箇所の特定ができます。

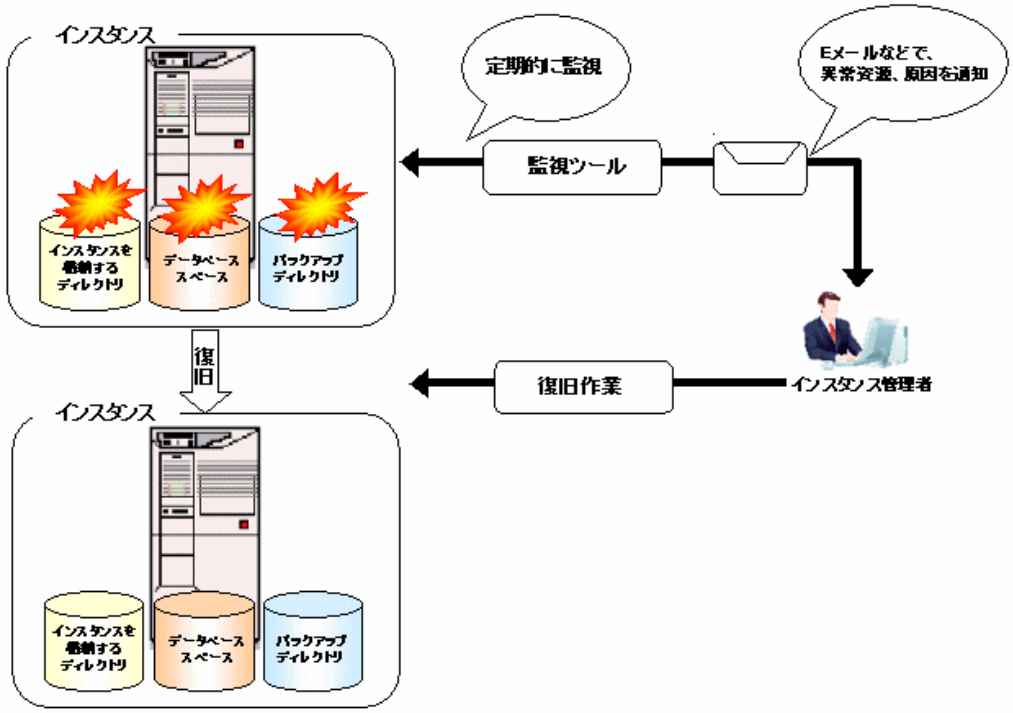
- データベースの異常
- バックアップの異常
- サーバ全体の異常

モニタリングでは、インスタンス資源の異常を早期に検出し、復旧することを目的にしています。PostgreSQL Plusでは、システムカタログに保持しているインスタンス資源のアクセス禁止状態の情報を監視することでモニタリングを実現できます。

モニタリングの運用

モニタリングの運用は、インスタンス資源の異常を監視し、異常を検出した場合には、インスタンス管理者に、インスタンスの異常から適切な復旧手段で復旧できる情報をメールなどで通知します。インスタンス資源の異常検出の通知を受けたインスタンス管理者は、適切な復旧作業を行います。

以下に、モニタリングの運用のイメージを示します。



異常検出時の復旧

モニタリングで検出したインスタンス資源の異常は、異常の資源や原因により復旧方法は異なり、適切な復旧作業を行う必要があります。以下に、モニタリングで検出できるおもなインスタンス資源の異常と復旧方法を示します。

データベースの異常

インスタンスを格納するディレクトリまたはデータベーススペースが破壊される異常です。メディアリカバリで復旧します。

バックアップの異常

バックアップディレクトリが破壊される異常です。バックアップディレクトリを再作成して復旧します。

サーバ全体の異常

インスタンスを格納するディレクトリまたはデータベーススペース、およびバックアップディレクトリの両方が破壊される異常です。ディザスタリカバリで復旧します。ディザスタリカバリで復旧するためには、ディザスタリカバリの運用を行う必要があります。

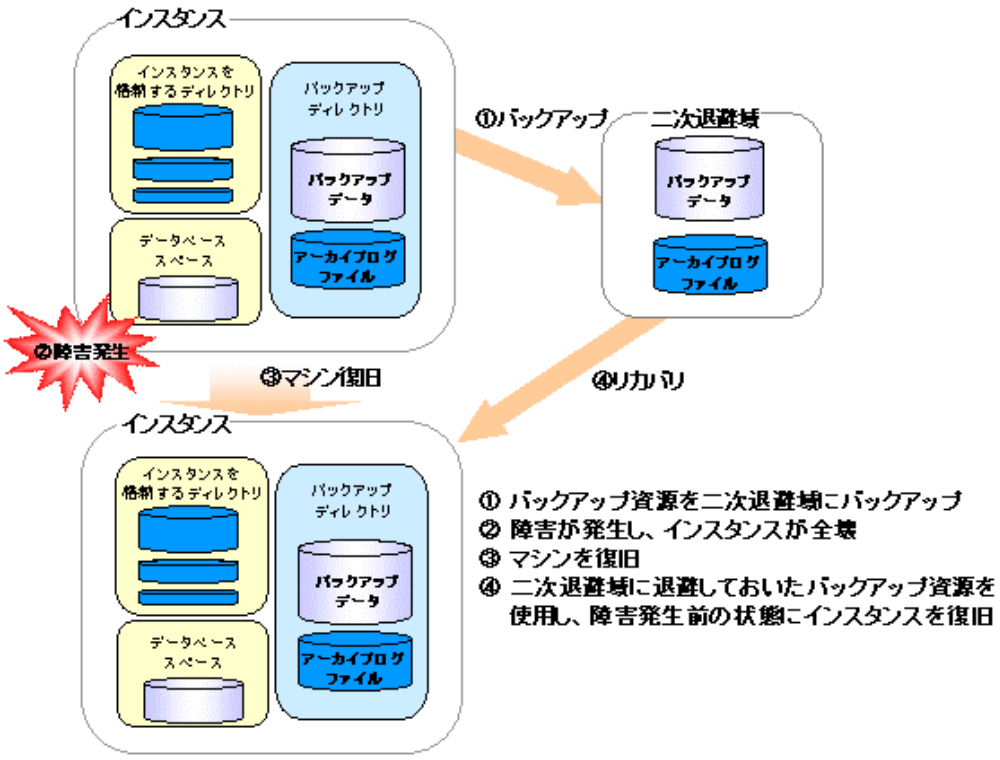
1.2 ディザスタリカバリ

ここでは、ディザスタリカバリの考え方について説明します。

ディザスタリカバリとは

ディザスタリカバリとは、災害発生などによりシステム全体に障害が発生した場合の、データベース復旧対策のことです。
PostgreSQL Plusでは、ディスク破壊などによって破損したデータベースを最新の状態にまで復旧する、メディアリカバリ機能を提供しています。システム全体に障害が発生した場合に備えて、メディアリカバリで使用するデータベースのバックアップデータとアーカイブログファイルを、二次媒体などに退避しておきます。この領域のことを二次退避域といいます。災害発生時には、二次退避域に退避しておいたデータベースのバックアップデータとアーカイブログファイルを使用して、データベースを復旧します。

以下に、PostgreSQL Plusのディザスタリカバリのイメージを示します。



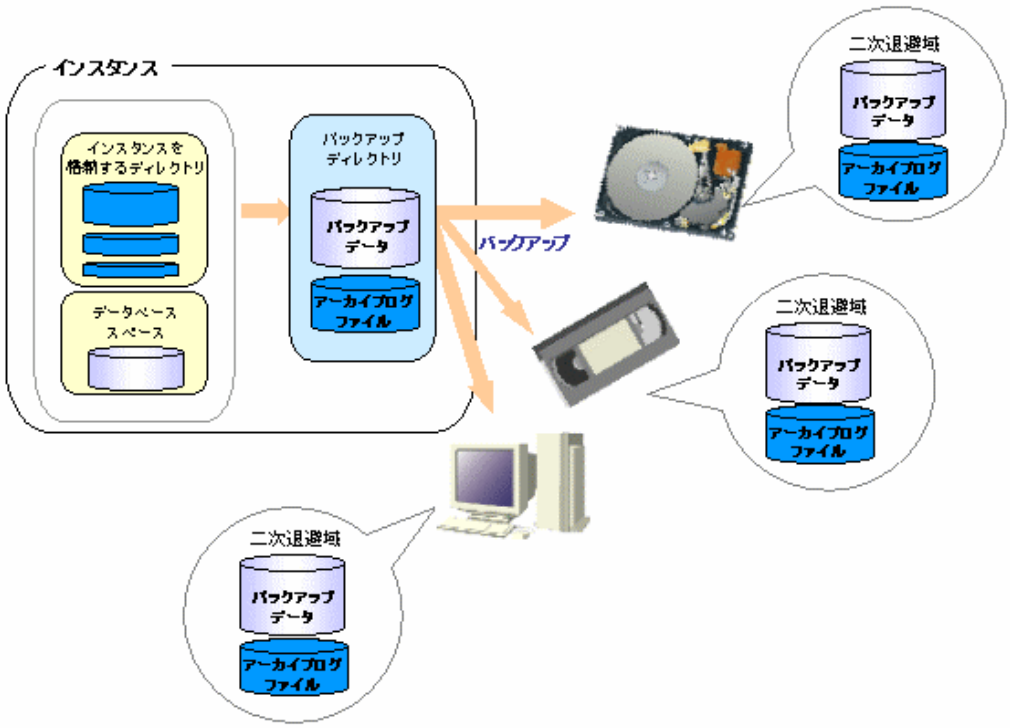
バックアップ

PostgreSQL Plusでは、インスタンスを格納するディレクトリが破壊されたときのメディアリカバリのために、更新の履歴を格納したアーカイブログファイルと、バックアップの実行によって作成されるバックアップデータをバックアップディレクトリに格納しています。
システム全体に障害が発生した場合に備えて、バックアップディレクトリに格納されているデータベースのバックアップデータとアーカイブログファイルを、OSのコマンドやバックアップソフトウェアを使用して二次退避域に退避しておきます。

二次退避域は、運用のコストや信頼性に合わせて自由に選択することができ、安全な場所へ持ち出すことが可能です。以下は、二次退避域の例です。

- ディスク
- MT装置
- 遠隔地の他のマシン

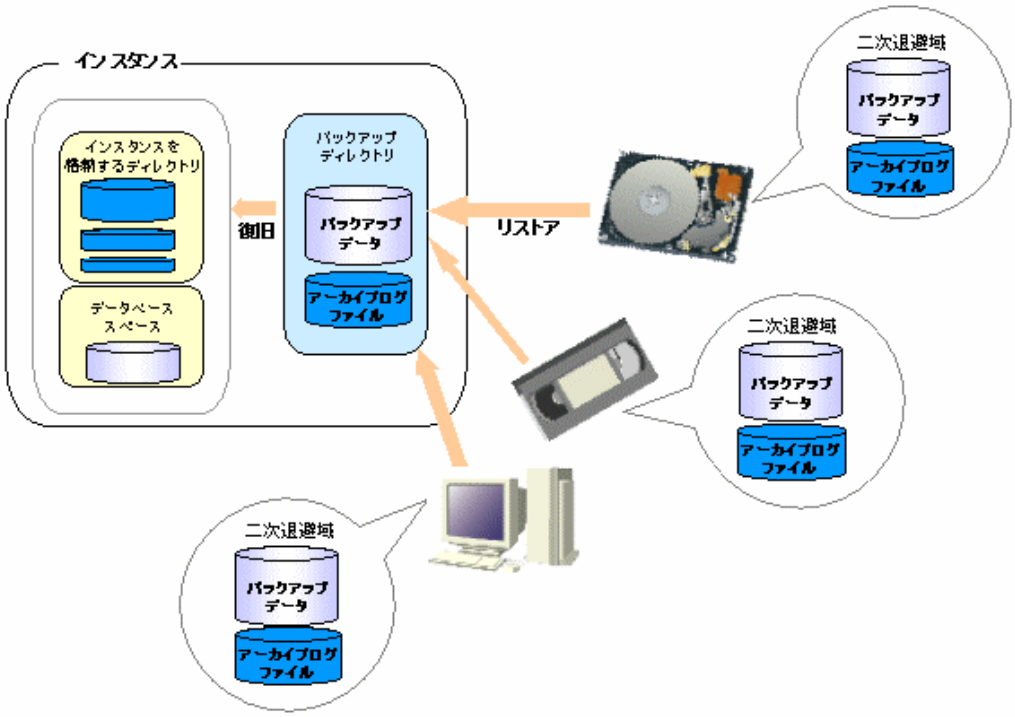
以下に、バックアップのイメージを示します。



リカバリ

リカバリは、二次退避域に退避しておいたデータベースのバックアップデータとアーカイブログファイルを、OSのコマンドやバックアップソフトウェアを使用して、バックアップディレクトリにリストアします。その後、メディアリカバリと同じ手順でデータベースを復旧します。

以下に、リカバリのイメージを示します。



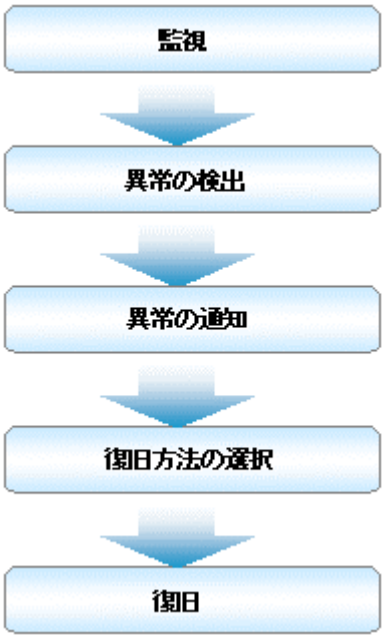
第2章　　運用設計

ここでは、モニタリングとディザスタリカバリの運用設計について説明します。

2.1 モニタリングの運用設計

ここでは、モニタリングの運用の流れに沿って、モニタリングの運用設計のポイントを説明します。

以下にモニタリングの運用の流れを示します。



2.1.1 監視

業務内容、データの重要度および異常が発生した場合の影響度を考慮して、監視契機、監視範囲を設計します。
例えば、業務で頻繁にアクセスされるデータベースに対しては、1時間間隔で監視し、業務であまりアクセスされないデータベースに対しては、1日1回監視するなどの設計を行います。

2.1.2 異常の検出

システムカタログから得られる各種資源のアクセス禁止状態を参照することで、異常を検出することができます。
アクセス禁止状態の資源名および原因を参照することで、異常となった

資源と異常の原因を知ることができます。

データベースの異常

データベースの異常の検出は、以下に示す資源のアクセス禁止状態で判断します。

- データベーススペース
- シーケンス
- システムカタログのテーブル、インデックス
- ユーザが作成したテーブル、インデックス

データベーススペース

pgx_dbspacesシステムカタログを参照して、データベーススペースの異常を検出します。

検出する異常	判断情報	
	status	cause
ハードウェア障害	INH	HER
ハードウェア以外の障害	INH	HER以外

シーケンス

pgx_sequencesシステムカタログを参照して、シーケンスの異常を検出します。

検出する異常	判断情報	
	status	cause
ハードウェア障害	INH	HER
ハードウェア以外の障害	INH	HER以外

システムカタログのテーブル、インデックス

pgx_tablesシステムカタログ、pgx_indexesシステムカタログを参照して、システムカタログのテーブル、インデックスの異常を検出します。

検出する異常	判断情報		
	schemaname	status	cause
ハードウェア障害	pg_catalog	INH	HER
ハードウェア以外の障害	pg_catalog	INH	HER以外

ユーザが作成したテーブル、インデックス

pgx_tablesシステムカタログ、pgx_indexesシステムカタログを参照して、ユーザが作成したテーブル、インデックスの異常を検出します。

検出する異常	判断情報		
	schemaname	status	cause
起動時の復旧異常	pg_catalog以外	INH	CRV
メディアリカバリの異	pg_catalog以外	INH	MRV

検出する異常	判断情報		
	schemaname	status	cause
常			
ロールバックの異常	pg_catalog以外	INH	TIH
高速コピーの失敗	pg_catalog以外	INH	ALD
ハードウェア障害	pg_catalog以外	INH	HER
上記以外の障害	pg_catalog以外	INH	上記以外

ユーザが作成したテーブル、インデックスは、データ破壊の他に、運用誤り、システム異常などで一時的にアクセス禁止状態となります。以下に一時的にアクセス禁止状態となる原因を示します。

起動時の復旧異常

以下に示すSQLコマンドを実行中にインスタンスがダウンした場合、起動時の復旧により対象資源を復旧できないため、アクセス禁止状態になります。

- COPYコマンド(高速コピーを利用した場合)
- CLUSTERコマンド
- TRUNCATEコマンド
- REINDEXコマンド
- DROP INDEXコマンド
- DROP TABLEコマンド
- DROP SCHEMA CASCADEコマンド

メディアリカバリの異常

以下に示すSQLコマンドを実行した状態で、バックアップを実行する前にリカバリを実行した場合、該当コマンドの対象となった資源はアクセス禁止状態になります。

- COPYコマンド(高速コピーを利用した場合)
- CLUSTERコマンド
- REINDEXコマンド
- CREATE INDEXコマンド

ロールバックの異常

以下に示すSQLコマンドは、ロールバックできません。ロールバックすると対象資源がアクセス禁止状態になります。また、ロールバック中にメモリ不足などのシステム異常が発生した場合もアクセス禁止状態になります。

- COPYコマンド(高速コピーを利用した場合)
- CLUSTERコマンド
- TRUNCATEコマンド
- REINDEXコマンド
- DROP INDEXコマンド
- DROP TABLEコマンド

● DROP SCHEMA CASCADEコマンド

高速コピーの失敗

高速コピーがエラーになった場合、ロールバックはされません。対象資源がアクセス禁止状態になります。

バックアップの異常

バックアップの異常の検出は、pgx_archive_logシステムカタログのアクセス禁止状態で判断します。

検出する異常	判断情報	状態
おもにハードウェア障害	status	inh

サーバ全体の異常

データベースとバックアップの両方の異常を検出した場合は、サーバ全体の異常と判断します。

2.1.3 異常の通知

異常を検出した場合に、適切な復旧方法を選択できる情報と、その情報を通知する手段を設計します。

通知内容

異常の資源の情報を通知し、さらに復旧手段を選択できる原因などの詳細情報を通知します。

● データベースの異常

— データベーススペース

pgx_dbspacesシステムカタログの以下の情報を通知します。

カラム名	情報	用途
dbspacename	データベーススペース名	業務への影響の判定
cause	原因	復旧方法の決定
device	デバイス名	異常媒体の確認

— シーケンス

pgx_sequencesシステムカタログの以下の情報を通知します。

カラム名	情報	用途
schemaname	スキーマ名	業務への影響の判定
sequencename	シーケンス名	業務への影響の判定
cause	原因	復旧方法の決定

— テーブル

pgx_tablesシステムカタログの以下の情報を通知します。

カラム名	情報	用途
schemaname	スキーマ名	業務への影響の判定
tablename	テーブル名	業務への影響の判定
tablespacename	割付先テーブルスペース名	異常媒体の確認
cause	原因	復旧方法の決定

— インデックス

pgx_indexesシステムカタログの以下の情報を通知します。

カラム名	情報	用途
schemaname	スキーマ名	業務への影響の判定
indexname	インデックス名	業務への影響の判定
tablespacename	割付先テーブルスペース名	異常媒体の確認
cause	原因	復旧方法の決定

● バックアップの異常

pgx_archive_logシステムカタログの以下の情報を通知します。

カラム名	情報	用途
archivepath	アーカイブパス名	異常媒体の確認

● サーバ全体の異常

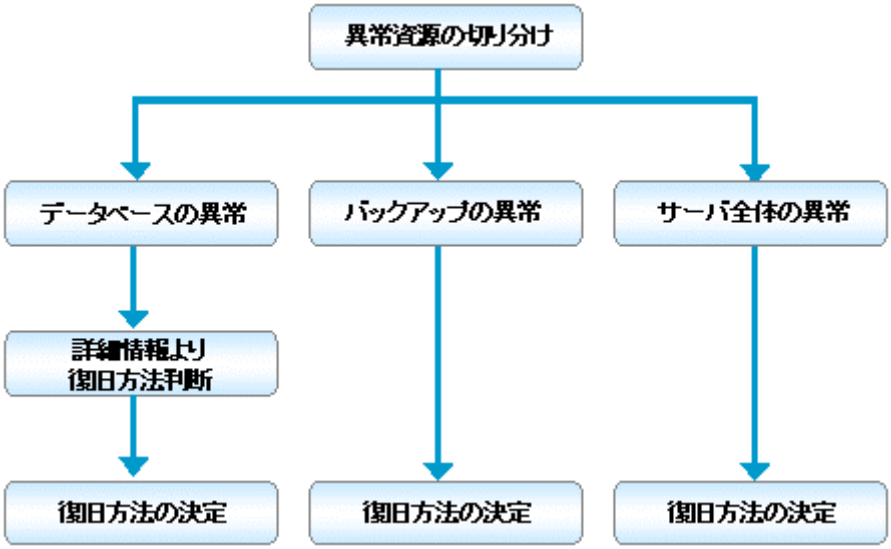
データベースの異常で通知する情報とバックアップの異常で通知する情報を通知します。

通知手段

メールを送信できる環境などでは、メールを使用して通知します。また、異常を通知する運用をすでに行っている場合は、その運用に連携するようにします。

2.1.4 復旧方法の選択

迅速に復旧作業が行えるよう、“2.1.5 復旧”で説明している復旧方法の手順を用意しておきます。以下に復旧方法を決定するまでの流れを示します。



2.1.5 復旧

異常からの復旧は、異常の資源、異常の原因により復旧方法が異なります。モニタリングで検出する異常ごとの復旧方法を説明します。

データベースの異常

データ破壊の場合はメディアリカバリで復旧します。それ以外は、異常の資源、異常の原因により復旧方法が異なります。以下に異常の資源、異常の原因ごとの復旧方法を説明します。
メディアリカバリでの復旧手順の詳細は、“PostgreSQL Plus 管理者ガイド”の“4.2 リカバリ”を参照してください。

データベーススペース

データベーススペースの異常からの復旧方法を以下に示します。

検出した異常	復旧方法
ハードウェア障害	ディスク交換後、メディアリカバリ
ハードウェア以外の障害	メディアリカバリ

シーケンス

シーケンスの異常からの復旧方法を以下に示します。

検出した異常	復旧方法
ハードウェア障害	ディスク交換後、メディアリカバリ
ハードウェア以外の障害	メディアリカバリ

システムカタログのテーブル、インデックス

システムカタログのテーブル、インデックスの異常からの復旧方法を以下に示します。

検出した異常	復旧方法
ハードウェア障害	ディスク交換後、メディアリカバリ
ハードウェア以外の障害	メディアリカバリ

ユーザが作成したテーブル、インデックス

ユーザが作成したテーブル、インデックスの異常からの復旧方法を以下に示します。

検出した異常	復旧方法
起動時の復旧異常	起動時の復旧異常からの復旧
メディアリカバリの異常	メディアリカバリの異常からの復旧
ロールバックの異常	ロールバックの異常からの復旧
高速コピーの失敗	高速コピーの失敗からの復旧
ハードウェア障害	ディスク交換後、メディアリカバリ
上記以外の障害	メディアリカバリ

起動時の復旧異常からの復旧

CLUSTERコマンド実行中にインスタンスがダウンしていた場合は、メディアリカバリで復旧します。CLUSTERコマンド以外のコマンドを実行中にインスタンスがダウンしていた場合は、実行していたSQLコマンドを再実行することで復旧します。以下に復旧手順を示します。

1. 原因の特定
   メッセージ“qdg71031u”または“qdg71033u”を参照し、SQLコマンドを実行中にインスタンスがダウンしたことを確認します。
2. SQLコマンドの再実行
   アクセス禁止状態になっている資源に対して実行したSQLコマンドを再実行します。

メディアリカバリの異常からの復旧

以下にメディアリカバリの異常からの復旧方法を示します。

COPYコマンド(高速コピーを利用した場合)およびCLUSTERコマンド:
バックアップ時点に復旧します。詳細は、“PostgreSQL Plus リファレンス”の“2.5 pgx_rcvall”を参照してください。

REINDEXコマンドおよびCREATE INDEXコマンド:
REINDEXコマンドなどでインデックスを再作成し復旧します。

ロールバックの異常からの復旧

CLUSTERコマンド実行中にインスタンスがダウンしていた場合は、メディアリ

カバリで復旧します。CLUSTERコマンド以外のコマンドを実行中にインスタンスダウンしていた場合は、実行していたSQLコマンドを再実行することで復旧します。ただし、実行していたSQLコマンドが特定できない場合は、メディアリカバリで復旧します。以下に復旧手順を示します。

1. 原因の特定
   メッセージ“qdg71031u” または“qdg71032u”を参照し、ロールバック不可能なSQLコマンドをロールバックしたことを確認します。

2. SQLコマンドの再実行
   アクセス禁止状態になっている資源に対して実行したSQLコマンドを再実行します。

高速コピーの失敗からの復旧

高速コピーの処理が失敗した原因を取り除いてから、高速コピーを再実行して復旧します。

バックアップの異常

バックアップの異常からの復旧は、バックアップディレクトリを再作成して復旧します。復旧手順の詳細は、“PostgreSQL Plus 管理者ガイド”の“4.2 リカバリ”を参照してください。

サーバ全体の異常

サーバ全体の異常からの復旧は、ディザスタリカバリで復旧します。ディザスタリカバリの運用手順は、“第4章 運用手順”を参照してください。

2.2 ディザスタリカバリの運用設計

ディザスタリカバリでは、障害発生時にどの状態まで復旧するかに合わせて、バックアップ運用の設計を行います。
ここでは、ディザスタリカバリのバックアップ運用の設計方法について説明します。

バックアップ運用は、以下の3種類のバックアップを組み合わせて設計します。

- 全体バックアップ
- 差分バックアップ
- 動作環境バックアップ

差分バックアップを使用する場合は、障害発生時にどの時点までデータベースを復旧する必要があるのかを判断して、以下の設計を行います。

- 差分バックアップの周期
- アーカイブログファイルの運用設計

バックアップの種類

バックアップの種類には、以下の3つがあります。

全体バックアップ

障害が発生した場合に、バックアップを行った時点まで復旧するための運用方法です。全体バックアップは、あらかじめ設計してあるバックアップ周期に合わせて定期的に取得したバックアップを、二次退避域に退避することで行います。

以下のコマンドを実行した場合には、バックアップ周期に関係なく、全体バックアップを実行します。

- COPYコマンド（SETコマンドにENABLE_DIRECT_COPYパラメータを指定して高速コピーを利用した場合）
- CLUSTERコマンド
- REINDEXコマンド
- CREATE INDEXコマンド
- CREATE DATABASEコマンド
- pgx_rcvallコマンド(-pオプション指定)

差分バックアップ

障害が発生した場合に、全体バックアップと差分バックアップを使用することにより、バックアップ周期内の最新の差分バックアップを取得した時

点にまで復旧するための運用方法です。差分バックアップを使用することで、障害発生時に損失するデータの量を少なくすることができます。差分バックアップは、全体バックアップ後にデータベースを更新した情報が格納されているアーカイブログファイルを、更新の差分データとして二次退避域に退避することで行います。

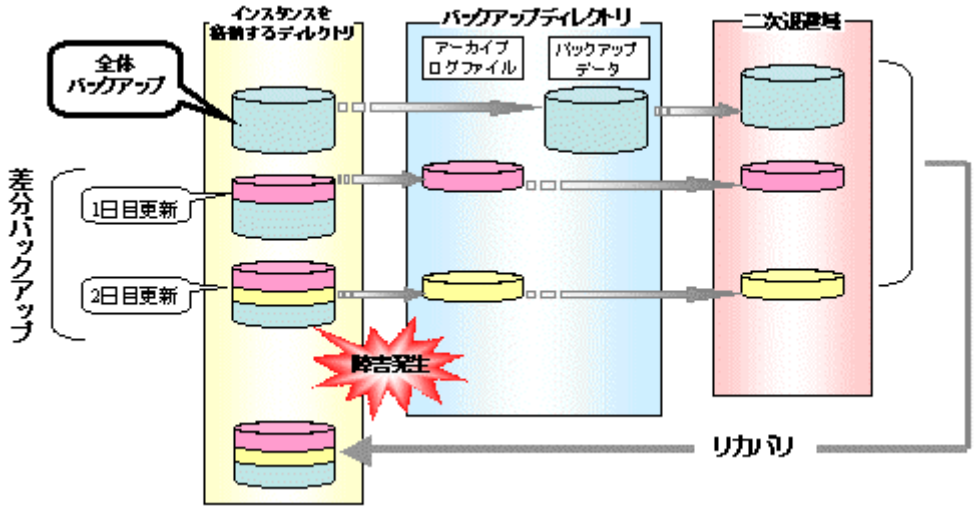
動作環境バックアップ

全体バックアップ後にメンテナンスをする場合など、動作環境の設定を変更した後に、各種設定ファイルを二次退避域に退避します。

以下のコマンドを実行して、動作環境の設定を変更した場合は、動作環境バックアップを実行します。

- PostgreSQL Plus Administratorまたは手動による動作環境の変更
- ユーザ追加/更新/削除

以下に、差分バックアップを使用したバックアップ運用の例を示します。以下の運用例では、1日前までデータベースを復旧する計画に沿って、毎日の業務終了後に差分バックアップを行う運用としています。



差分バックアップの運用設計

差分バックアップを使用するには、以下の2つの設計を行います。

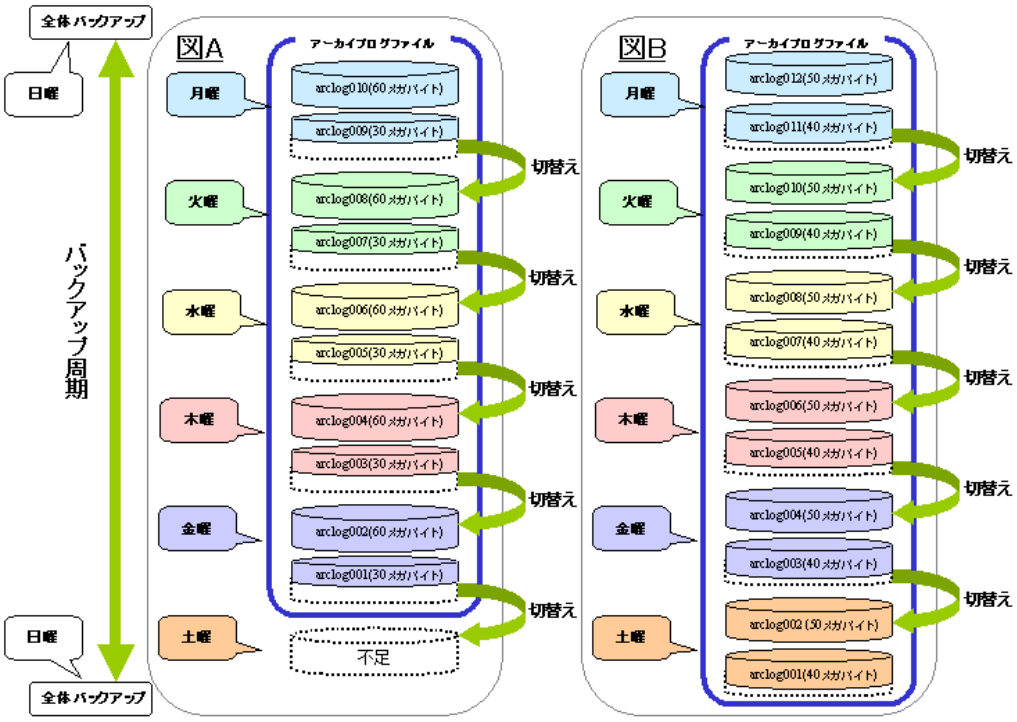
差分バックアップの周期

バックアップ周期内で差分バックアップを行う周期を設計します。復旧したい時点に合わせて、差分バックアップの周期を設計してください。
たとえば、前日の状態まで復旧したい場合は、差分バックアップの周期を1日として、バックアップ周期内で毎日差分バックアップを行うように設計します。

アーカイブログファイルの運用設計

差分バックアップの周期に合わせて、アーカイブログファイルの数を設計します。
差分バックアップを行う場合、使用中のアーカイブログファイルを強制的に切り替えます。これにより、差分バックアップを行うアーカイブログファイルに対して更新ログがとられなくなるため、復旧するポイントが確定されます。強制的にアーカイブログファイルを切り替えると、それまで使用していたアーカイブログファイルに空き容量があっても、その分は使用されなくなります。アーカイブログファイルのサイズはバックアップ周期に合わせて設定しています。しかし、初期設定ではアーカイブログファイルの数を10個で構成しているので、バックアップ周期内の差分バックアップ回数が多い場合は、アーカイブログファイルの数が不足することがあります。
バックアップ周期内でアーカイブログファイルが不足することのないように、差分バックアップの周期と毎回の差分バックアップの量に合わせて、アーカイブログファイルを設計してください。
アーカイブログファイルが10個では足りない場合は、アーカイブログファイルを必要な数だけ再作成します。

たとえば以下の図は、全体バックアップを毎週日曜に行い、それに合わせて差分バックアップを月曜から土曜の毎日行う場合の、アーカイブログファイルの運用設計例です。この例では、1 週間のアーカイブログファイルの使用量を540メガバイト、1日の使用量を平均90メガバイト、アーカイブログファイルのサイズを600メガバイトと設定しています。
図Aのように、アーカイブログファイルの数が初期設定の10個の場合は、アーカイブログファイルの切替え作業によって、アーカイブログファイルに使用されない空き容量ができてしまうため、土曜のデータベース更新情報を格納するためのアーカイブログファイルが不足してしまいます。
そこで、図Bのようにアーカイブログファイルの数を12個にして、1個のアーカイブログファイルのサイズを小さくし、アーカイブログファイルの切替え作業によって使用されなくなってしまう部分を減らすことで、問題なくバックアップ運用を行うことができます。


全体バックアップ
日曜
バックアップ周期
日曜
全体バックアップ
図A
アーカイブログファイル
月曜
火曜
水曜
木曜
金曜
土曜
arclog010(60メガバイト)
arclog009(30メガバイト)
arclog008(60メガバイト)
arclog007(30メガバイト)
arclog006(60メガバイト)
arclog005(30メガバイト)
arclog004(60メガバイト)
arclog003(30メガバイト)
arclog002(60メガバイト)
arclog001(30メガバイト)
不足
切替え
切替え
切替え
切替え
切替え
図B
アーカイブログファイル
月曜
火曜
水曜
木曜
金曜
土曜
arclog012(50メガバイト)
arclog011(40メガバイト)
arclog010(50メガバイト)
arclog009(40メガバイト)
arclog008(50メガバイト)
arclog007(40メガバイト)
arclog006(50メガバイト)
arclog005(40メガバイト)
arclog004(50メガバイト)
arclog003(40メガバイト)
arclog002(50メガバイト)
arclog001(40メガバイト)
切替え
切替え
切替え
切替え
切替え

第3章　　運用の環境設定

ここでは、具体的な運用の環境設定手順を説明します。

3.1 モニタリング機能の環境設定

スクリプトを作成することにより、モニタリングを自動化することができます。ここでは、モニタリング用のスクリプトの作成とモニタリングを自動化する方法について、説明します。
モニタリングでは、データベースサーバへ同時に接続するアプリケーション多重度を1つ増やす必要があります。アプリケーションとの通信に関する設定内容については、“PostgreSQL Plus 管理者ガイド”の“3.5.4 アプリケーションとの通信の設定”を参照してください。

スクリプトファイルの作成

モニタリングのスクリプトには、以下の処理が必要になります。

- 異常の検出
  “2.1 モニタリングの運用設計”の“異常の検出”を参考にし、システムカタログを参照して検出します。
- 異常の通知
  “2.1 モニタリングの運用設計”の“異常の通知”を参考にし、インスタンス管理者にメールなどで異常を通知します。

サンプルスクリプトについては、“B.1 モニタリングのサンプルスクリプト”を参照してください。

モニタリングの自動化の方法

モニタリング用のプログラムが定期的に実行されるようにスケジュールする方法を説明します。

cronを使用してモニタリング用のスクリプトなどを定期的に実行されるようにスケジュールします。以下にcronを使用してモニタリング用のサンプルプログラムをスケジュールする手順を示します。

1. cronのデーモン“crond”の状態確認
   以下のコマンドは、スーパーユーザーで実行します。

```
# /etc/rc.d/init.d/crond status
crond (pid 917) を実行中...
```

   crondが起動していなかった場合にはcrondを手動で起動してください。また、OS起動時にcrondが自動起動するように設定してください。詳細は、OSのマニュアルを参照してください。

2. crontabファイルの編集
   以下のコマンドを実行して、crontabファイルを編集します。
   以下のコマンドは、インスタンス管理者用ユーザで実行します。

```
$ crontab -e
```

crontabファイルは、以下の書式で設定します。

```
min hour day month week command
```

min　　：分を指定する。
hour　：時を指定する。
day　　：日を指定する。
month　：月を指定する。
week　　：曜日を指定する。日曜日を0とし，順に数字に当てはめていく。
command：実行するコマンドまたはスクリプトを記述する。

以下は、毎日9時から20時の間、1時間間隔で監視プログラムが実行されるように設定したcrontabファイルの内容です。

```
# 毎日　9:00 - 20:00 1時間間隔で監視
0 9-20/1 * * * $HOME/bin/monitoring.sh
```

ここでは、crontabコマンドを使用した手順を記載していますが、他の方法もあります。crontabコマンドの詳細およびcrontabコマンドを使用しない場合の手順については、OSのマニュアルを参照してください。

3.2 ディザスタリカバリの環境設定

バックアップの運用設計でアーカイブログファイルの再作成が必要な場合は、以下の手順で再作成します。
手順内の実行例では、インスタンスを格納するディレクトリを/PlusData/data1、バックアップディレクトリを/PlusBackup/data1としています。

アーカイブログファイルの再作成手順

以下の手順は、インスタンス管理者用ユーザで実行します。PostgreSQL Plusは停止した状態で行ってください。

1. 既存アーカイブログファイルの削除
   pgx_arclogコマンドのdropモードを実行し、バックアップディレクトリ配下にarclog001～arclog010までの連番で存在する、10個のアーカイブログファイルを削除します。

```
$ pgx_arclog drop -D /PlusData/data1  /PlusBackup/data1/arclog001
$ pgx_arclog drop -D /PlusData/data1  /PlusBackup/data1/arclog002
                                   :
$ pgx_arclog drop -D /PlusData/data1  /PlusBackup/data1/arclog010
```

2. アーカイブログファイルの作成
   アーカイブログファイルの作成は、1個のアーカイブログファイルのサイズを指定してバックアップディレクトリ配下に初期作成し、それに対して必要な数のアーカイブログファイルを追加して行います。初期作成で指定するアーカイブログファイルは、設計したサイズを指定してください。追加で作成されるアーカイブログファイルのサイズは、初期作成で指定したサイズになります。
   アーカイブログファイルのファイル名は、“arclog001”というように、ファイル名の先頭を“arclog”とし、001から連番で指定してください。
   ここでは、バックアップディレクトリ配下に、arclog001～arclog012までの12個のアーカイブログファイルを作成します。

   — アーカイブログファイルの初期作成
   アーカイブログファイルの初期作成は、pgx_arclogコマンドのcreateモードで行います。

```
$ pgx_arclog create -D /PlusData/data1  /PlusBackup/data1/arclog001  50M
```

   — アーカイブログファイルの追加

アーカイブログファイルの初期作成が終わったら、pgx_arclogコマンドのaddモードでアーカイブログファイルを必要な数だけ追加します。

```
$ pgx_arclog add -D /PlusData/data1  /PlusBackup/data1/arclog002
$ pgx_arclog add -D /PlusData/data1  /PlusBackup/data1/arclog003
                                  :
$ pgx_arclog add -D /PlusData/data1  /PlusBackup/data1/arclog012
```

3. バックアップの実行
   インスタンスを起動し、バックアップを実行します。
   バックアップの手順については、“PostgreSQL Plus 管理者ガイド”の“4.1 バックアップ”を参照してください。

アーカイブログファイルのサイズをPostgreSQL Plus Administratorから変更すると、アーカイブログファイルの数は初期値の10個に戻ってしまいます。
アーカイブログファイルのサイズを変更する場合は、上記の手順どおりにコマンドを使用して、新しいサイズでアーカイブログファイルを再作成してください。

第4章　運用手順

ここでは、具体的なディザスタリカバリのバックアップ手順とリカバリ手順を説明します。

手順内の実行例では、以下のディレクトリを使用しています。

インスタンスを格納するディレクトリ	/PlusData/data1
バックアップディレクトリ	/PlusBackup/data1
二次退避域のディレクトリ	/PlusArchive/data1
データベーススペース	/PlusDbspace/data1

ディザスタリカバリのサンプルスクリプトについては、“B.2 ディザスタリカバリのサンプルスクリプト”を参照してください。

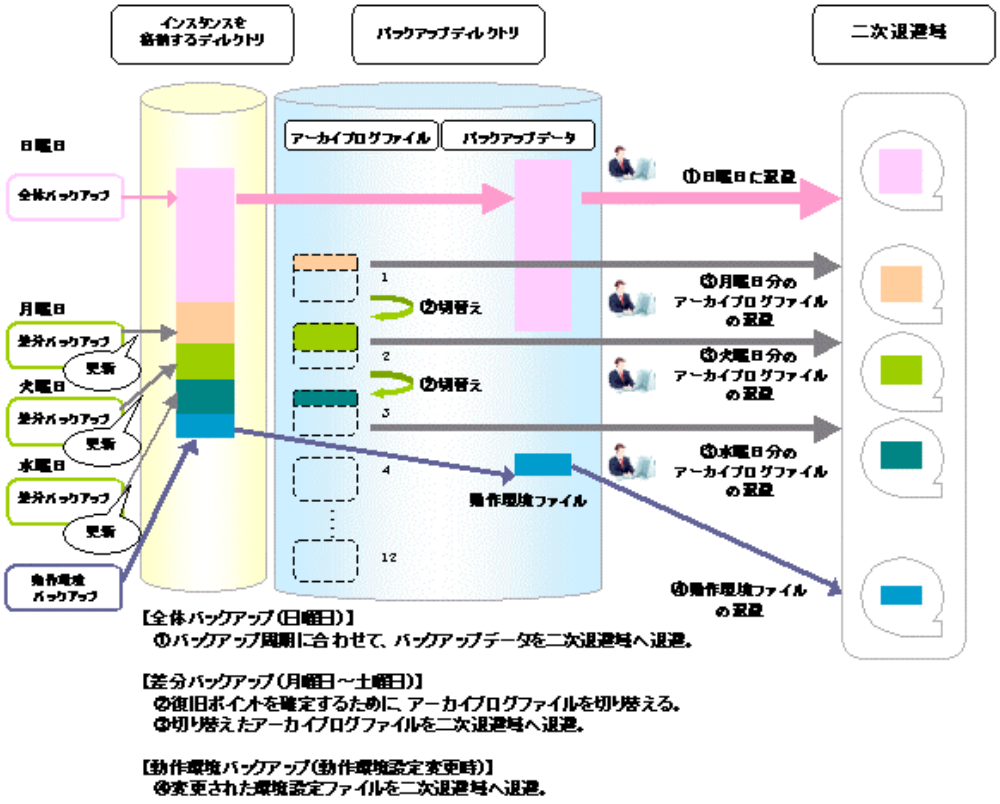
4.1 バックアップ手順

バックアップの種類には、以下の3つがあります。

- 全体バックアップ
- 差分バックアップ
- 動作環境バックアップ

バックアップデータとアーカイブログファイルの二次退避域への退避には、OSのコピーコマンドまたはバックアップソフトウェアを使用します。

以下の図は、日曜日に全体バックアップを行い、月曜日から土曜日は差分バックアップを行う場合の、バックアップのイメージです。



4.1.1 全体バックアップの手順

以下に、全体バックアップの手順を示します。
以下の手順は、インスタンス管理者用ユーザで実行します。

1. インスタンス全体のバックアップ取得
   バックアップの手順については、“PostgreSQL Plus 管理者ガイド”の“4.1 バックアップ”を参照してください。

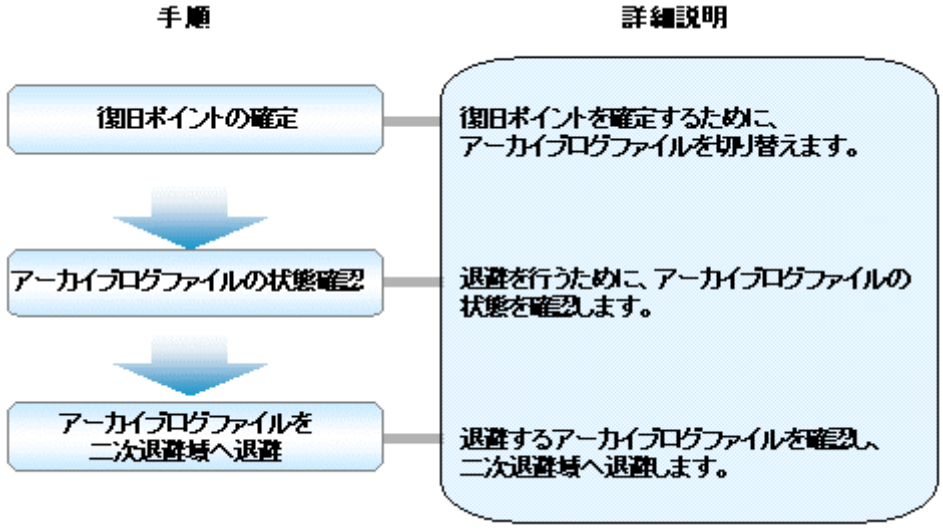
2. バックアップデータだけを二次退避域へ退避
   バックアップディレクトリ配下にあるbkup1ディレクトリをすべて二次退避域に退避します。

   以下に、OSのコピーコマンドを使った実行例を示します。

```
$ cp -rf /PlusBackup/data1/bkup1  /PlusArchive/data1
```

4.1.2 差分バックアップの手順

以下に、差分バックアップの手順を示します。
以下の手順は、インスタンス管理者用ユーザで実行します。



1. 復旧ポイントの確定
   現在使用しているアーカイブログファイルを切り替えて、復旧ポイントを確定します。
   アーカイブログファイルの切替えには、pgx_arclogコマンドのswitchモードを使用します。

```
$ pgx_arclog switch -D /PlusData/data1
INFO: qdg03800i:アーカイブログファイルを切り替えました ファイ
```

```
ル名='/PlusBackup/data1/arclog010'
通番='3'
```

pgx_arclogコマンドのswitchモードを実行して以下のエラーメッセージが出力された場合には、アーカイブログファイルが不足しています。
pgx_arclogコマンドのaddモードで空きのアーカイブログファイルを用意し、pgx_arclogコマンドのswitchモードを再実行してください。また、今後のアーカイブログファイルの不足を防ぐために、アーカイブログファイルの設計の見直しを実施してください。

qdg02437u:アーカイブログファイルの強制切換え処理は空きのアーカイブログファイルがないため処理できません

2. アーカイブログファイルの状態確認
差分バックアップの対象となるアーカイブログファイルを確認するために、アーカイブログファイルの状態を確認します。
アーカイブログファイルの状態を確認するには以下の2つの方法があります。いずれかの方法でアーカイブログファイルの状態を確認してください。
   - コマンドで確認する方法
   - psqlを使用してシステムカタログで確認する方法

アーカイブログファイルの状態は、出力された情報の“status”を見ることで確認できます。“status”の説明については、“表:アーカイブログファイルの状態”を参照してください。

   a) コマンドで確認する方法
   pgx_arclogコマンドのviewモードを実行することで、アーカイブログファイルの状態を確認できます。

```
$ pgx_arclog view -D /PlusData/data1
serial# status   total    used     avail    capacity backup-date path
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog001
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog002
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog003
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog004
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog005
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog006
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog007
      0  empty    50M      0        50M         0%     ------   /PlusBackup/data1/arclog008
```

```
        0  empty    50M     0       50M     0%      ------  /PlusBackup/data1/arclog009
        3  using    50M     0       50M     0%      ------  /PlusBackup/data1/arclog010
        2  switch   50M     5240k   45960k  10%     ------  /PlusBackup/data1/arclog011
        1  full     50M     50M     0       100%    ------  /PlusBackup/data1/arclog012
```

b) psqlを使用してシステムカタログで確認する方法
psqlを使用してpgx_archive_logシステムカタログを参照することで、アーカイブログファイルの状態を確認できます。

```
postgres=# select * from pgx_archive_log ;
groupname |       archivepath          |serialno|   size   | usedsize |backuptime| status
----------+----------------------------+--------+----------+----------+----------+-------
system    |/PlusBackup/data1/arclog001|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog002|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog003|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog004|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog005|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog006|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog007|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog008|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog009|      0 | 52428800 |        0 |          | empty
system    |/PlusBackup/data1/arclog010|      3 | 52428800 |        0 |          | using
system    |/PlusBackup/data1/arclog011|      2 | 52428800 |  5300224 |          | switch
system    |/PlusBackup/data1/arclog012|      1 | 52428800 | 52363264 |          | full
(12 rows)
```

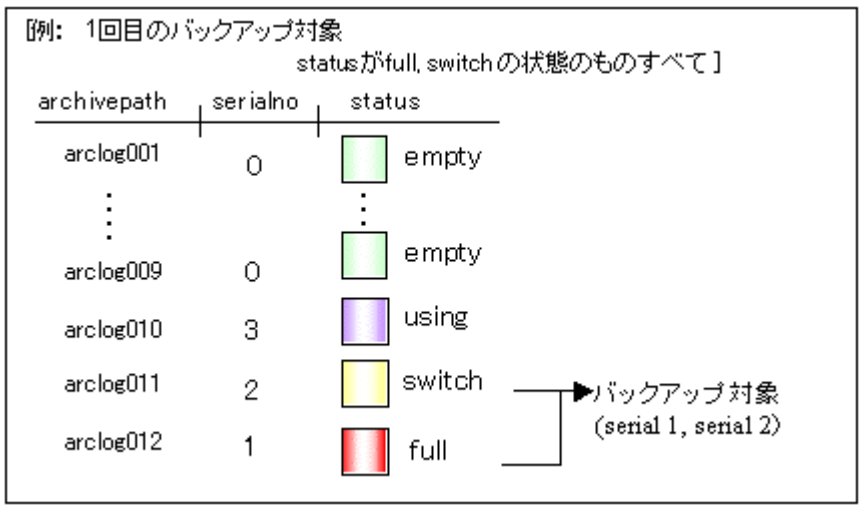
アーカイブログファイルの状態

アーカイブログファイルの状態(status)	説明	退避の必要性
using	現在使用中のアーカイブログファイル。 実行されるトランザクションの更新情報が格納される。	無し
full	更新情報が満杯。	有り
empty	未使用。 今後使用されるアーカイブログファイル。	無し
inh	異常発生による閉塞状態。 アーカイブログファイルが破損している状態。(注)	無し

アーカイブログファイルの状態(status)	説明	退避の必要性
switch	使用中のアーカイブログファイルを強制的に切り替えた状態。 切替え前に蓄積された更新情報が格納されている。	有り
icoff	pgx_dmpallコマンドでアーカイブログファイルを破棄した状態。 今後使用されるアーカイブログファイル。	無し

注)バックアップディスク異常ですので、ディスクの交換を行い復旧してください。復旧手順は、“PostgreSQL Plus 管理者ガイド”の“4.2 リカバリ”を参照してください。

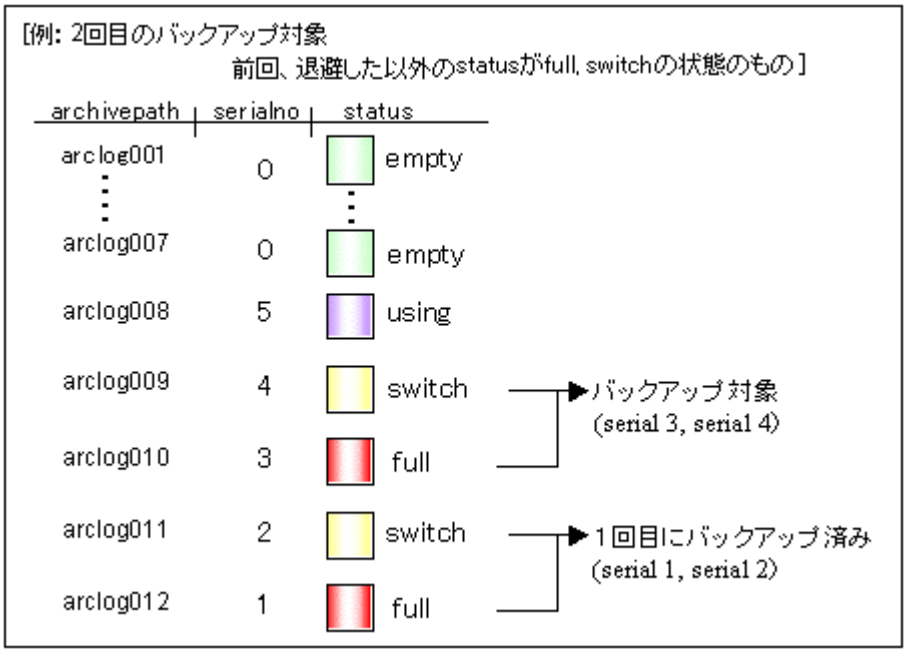
3. アーカイブログファイルを二次退避域へ退避
二次退避域へ退避するアーカイブログファイルは、データベースの更新情報が蓄積されたアーカイブログファイルだけになります。退避するアーカイブログファイルを確認し、退避する必要のあるアーカイブログファイルを二次退避域へ退避します。退避するとき、アーカイブログファイルの名前は変更しないでください。なお、全体バックアップ以降初めて差分バックアップを行う場合と、2回目以降の場合とで退避するアーカイブログファイルが異なります。以下にその説明をします。
   a) 全体バックアップ以降初めての差分バックアップを行う場合
   “表:アーカイブログファイルの状態”にある退避の必要性“有り”の状態となっているすべてのアーカイブログファイルを退避します。



以下に、OSのコピーコマンドを使って、二次退避域のディレクトリ配下にある1stディレクトリに退避する実行例を示します。

```
$ cp -f /PlusBackup/data1/arclog012  /PlusArchive/data1/1st
$ cp -f /PlusBackup/data1/arclog011  /PlusArchive/data1/1st
```

b) 全体バックアップ後、2回目以降の差分バックアップを行う場合“表:アーカイブログファイルの状態”にある退避の必要性“有り”のアーカイブログファイルのうち、シリアル番号(pgx_arclogコマンドのviewモードで表示される“serial#”)が前回の差分バックアップで退避したファイルの次のものから、今回切り替えたファイルまでの間のものを退避します。使用されたアーカイブログファイルには、使用した順にシリアル番号がふられています。



以下に、OSのコピーコマンドを使って、全体バックアップ以降2回目の差分バックアップを、二次退避域のディレクトリ配下にある2ndディレクトリに退避する実行例を示します。

```
$ cp -f /PlusBackup/data1/arclog010  /PlusArchive/data1/2nd
$ cp -f /PlusBackup/data1/arclog009  /PlusArchive/data1/2nd
```

4.1.3　　動作環境バックアップの手順

以下に、動作環境バックアップの手順を示します。
以下の手順は、インスタンス管理者用ユーザで実行します。

1. 各種設定ファイルだけをバックアップ
   pgx_dmpallコマンドの-cオプションを実行することで、各種設定ファイルだけがバックアップされます。

```
$ pgx_dmpall -D /PlusData/data1 -c
```

2. 各種設定ファイルを二次退避域へ退避
   バックアップディレクトリ配下のbkup1ディレクトリにあるファイルのうち、ファイル名の先頭文字が“#”でないファイルを、二次退避域に退避してください。

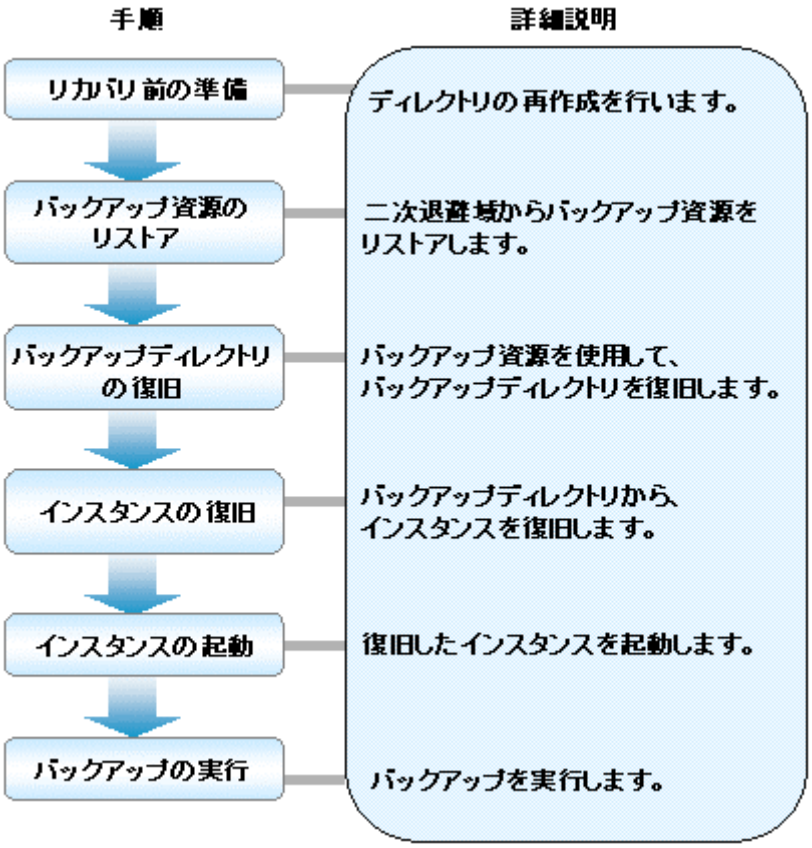
   以下に、OSのコピーコマンドを使って二次退避域のディレクトリ配下にあるconfディレクトリに退避する実行例を示します。

```
$ cp -f /PlusBackup/data1/bkup1/ファイル名　/PlusArchive/data1/conf
```

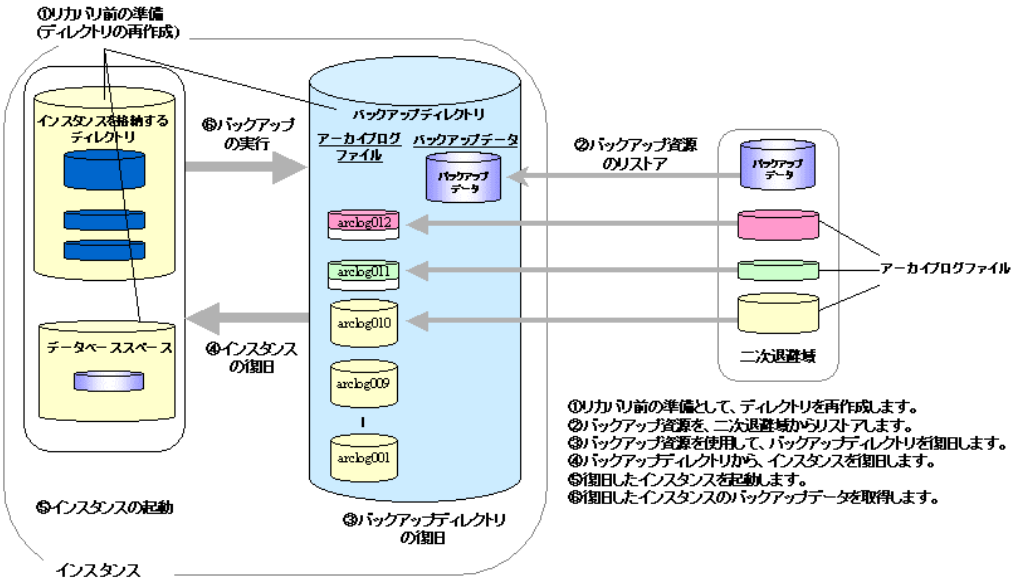
4.2　リカバリ手順

システム全体の障害が発生した場合は、二次退避域に退避しておいたバックアップ資源を使用してバックアップディレクトリを復旧し、その後メディアリカバリと同じ手順でインスタンスを格納するディレクトリを復旧します。

リカバリは、以下の手順で行います。



リカバリ手順のイメージを、以下に示します。



リカバリ前の準備

リカバリ前の準備について説明します。

以下の手順は、スーパーユーザー(root)で行います。

1. ディレクトリの再作成
   データベースを復旧するために必要なディレクトリを作成します。
   障害発生前と同じ構成で、以下のディレクトリを作成してください。

   - インスタンスを格納するディレクトリ
   - バックアップディレクトリ
   - データベーススペース

   以下に実行例を示します。

```
# mkdir -p  /PlusData/data1
# mkdir -p  /PlusBackup/data1
# mkdir -p  /PlusDbspace/data1
```

2. ディレクトリの権限の設定
   作成したディレクトリを、インスタンス管理者ユーザだけが書き込み可能な権限にしておきます。
   ディレクトリの権限をインスタンス管理者用ユーザ“postgres”に変更する実行例を、以下に示します。

```
# chown postgres  /PlusData/data1  /PlusBackup/data1  /PlusDbspace/data1
# chgrp postgres  /PlusData/data1  /PlusBackup/data1  /PlusDbspace/data1
# chmod 700  /PlusData/data1  /PlusBackup/data1  /PlusDbspace/data1
```

バックアップ資源のリストア

二次退避域に退避していたバックアップ資源を、リカバリ前の準備で作成したバックアップディレクトリへリストアします。
以下の手順は、インスタンス管理者用ユーザで実行します。

1. バックアップデータのリストア
   バックアップデータであるbkup1ディレクトリを、二次退避域からバックアップディレクトリにコピーします。

```
$ cp -rf  /PlusArchive/data1/bkup1  /PlusBackup/data1
```

2. バックアップデータを格納するbkup1ディレクトリの登録
   バックアップディレクトリ配下にbackupという名前でbkup1ディレクトリへのシンボリックリンクを作成します。

```
$ ln -s bkup1 /PlusBackup/data1/backup
```

3. アーカイブログファイルのリストア
   差分バックアップを実行していた場合に行います。
   二次退避域に退避したアーカイブログファイルをバックアップディレクトリにコピーします。二次退避域の退避ディレクトリ配下に1st、2ndとある場合の実行例を以下に示します。

```
$ cp -f /PlusArchive/data1/1st/arclog*  /PlusBackup/data1
$ cp -f /PlusArchive/data1/2nd/arclog*  /PlusBackup/data1
```

4. 各種設定ファイルだけをリストア
   動作環境バックアップを実行していた場合に行います。
   二次退避域に退避した各種設定ファイルをbkup1ディレクトリ配下にコピーします。二次退避域の退避ディレクトリ配下conf内に、各種設定ファイルを退避している場合の実行例を以下に示します。

```
$ cp -f  /PlusArchive/data1/conf/* /PlusBackup/data1/bkup1
```

バックアップディレクトリの復旧

二次退避域に退避していなかった分のアーカイブログファイルを再作成し、バックアップディレクトリを復旧します。
まず、アーカイブログファイルの再作成に必要な環境を作成し、その後、アーカイブログファイルの再作成を行います。
以下の手順は、インスタンス管理者用ユーザで実行します。

1. 各種設定ファイルの準備
   バックアップディレクトリのbkup1ディレクトリ配下に格納されている以下の各種設定ファイルを、インスタンスを格納するディレクトリ配下にコピーしてください。
   - edmf.conf
   - edmfpool.conf
   - postgresql.conf

```
 $ cp -f /PlusBackup/data1/bkup1/edmf.conf  /PlusData/data1
$ cp -f /PlusBackup/data1/bkup1/edmfpool.conf  /PlusData/data1
$ cp -f /PlusBackup/data1/bkup1/postgresql.conf  /PlusData/data1
```

2. インスタンスを格納するディレクトリにバックアップディレクトリを登録
   インスタンスを格納するディレクトリ配下にbackupdirという名前でバックアップディレクトリへのシンボリックリンクを作成します。

```
  $ ln -s /PlusBackup/data1 /PlusData/data1/backupdir
```

3. インスタンスを格納するディレクトリにログ情報を登録
   PostgreSQL Plusのログ情報を、インスタンスを格納するディレクトリに登録します。ログ情報の登録には、pgx_ctllogコマンドのcreateモードを使用します。

```
$ pgx_ctllog create -D /PlusData/data1
```

END:"#Linux"

4. テンポラリログの作成
   テンポラリログを作成します。
   ここで作成するテンポラリログは、一時的なものです。テンポラリログはメディアリカバリで復旧されますので、ここでは以下の実行例の値で作成してください。
   pgx_tlogディレクトリとTLF.logファイルの名前は変更しないでください。まず、pgx_tlogディレクトリをインスタンスを格納するディレクトリに作成し、インスタンス管理者用ユーザだけが書き込み可能な権限に変更します。その後、pgx_tmplogコマンドのcreateモードでテンポラリログを作成します。

```
 $ mkdir /PlusData/data1/pgx_tlog
 $ chmod 700 /PlusData/data1/pgx_tlog
 $ pgx_tmplog create -D /PlusData/data1 -c 64 /PlusData/data1/pgx_tlog/TLF.log -in
 -in 50M 20M 64
```

5. アーカイブログファイルの作成
   バックアップディレクトリに存在しないアーカイブログファイルを作成します。

差分バックアップを行っている場合と行っていない場合とで、作成するアーカイブログファイルの数が異なります。

— 差分バックアップを行っている場合
ここでは、1個のアーカイブログファイルのサイズを50メガバイトで12個(arclog001～arclog012)作成して運用し、差分バックアップで退避したアーカイブログファイルをarclog009～arclog012とした場合を例にして説明します。

a- 運用設計時に設定したアーカイブログファイルのうち、リストアしていないアーカイブログファイルを確認します。

b- リストアしていないアーカイブログファイル(arclog001～arclog008)のうち、1個をpgx_arclogコマンドのcreateモードで、アーカイブログファイルのサイズを指定して、初期作成します。リストアしたアーカイブログファイルがあっても、初期作成は必ず行います。

```
$ pgx_arclog create -D /PlusData/data1  /PlusBackup/data1/arclog001  50M
```

c- アーカイブログファイルの初期作成が終わったら、pgx_arclogコマンドのaddモードで残りのアーカイブログファイル(arclog002～arclog008)を追加作成します。

```
$ pgx_arclog add -D /PlusData/data1  /PlusBackup/data1/arclog002
$ pgx_arclog add -D /PlusData/data1  /PlusBackup/data1/arclog003
                          :
$ pgx_arclog add -D /PlusData/data1  /PlusBackup/data1/arclog008
```

— 差分バックアップを行っていない場合
“3.2 ディザスタリカバリの環境設定”の“アーカイブログファイルの再作成手順”の“手順2.”を参照して、運用設計時に設定した数のアーカイブログファイルをすべて作成してください。

インスタンスの復旧

pgx_rcvallコマンドを利用し、インスタンスを格納するディレクトリおよびデータベーススペースを復旧します。
差分バックアップを使用する場合と使用しない場合とで実行するコマンドが異なりますので、注意してください。
インスタンスの復旧は、インスタンス管理者用ユーザで実行してください。

- 差分バックアップを使用してリカバリする場合
  バックアップデータとアーカイブログファイルを使用してインスタンスを復旧します。

```
$ pgx_rcvall -D /PlusData/data1 -B /PlusBackup/data1
```

- 差分バックアップを使用しないでリカバリする場合
  バックアップ時点への復旧になりますので、pgx_rcvallコマンドに-pオプションが必要です。

```
$ pgx_rcvall -D /PlusData/data1 -B /PlusBackup/data1 -p
```

インスタンスの起動

インスタンスを起動します。

バックアップの実行

バックアップを実行します。
バックアップの手順については、“PostgreSQL Plus 管理者ガイド”の“4.1 バックアップ”を参照してください。

付録A コマンドリファレンス

ここでは、本書内で使用されている以下のコマンドについて説明します。
以下のコマンドを実行する場合、インスタンス管理者用ユーザで実行してください。

- pgx_arclog
- pgx_ctllog
- pgx_tmplog

1.1 pgx_arclog

Name

pgx_arclog - アーカイブログファイルの操作

Synopsis

```
pgx_arclog create [-D datadir] ALFname ALFsize
pgx_arclog add [-D datadir] ALFname
pgx_arclog drop [-D datadir] ALFname
pgx_arclog view [-D datadir] [ALFname]
pgx_arclog backup [-D datadir] ALFbackupname@devicename
pgx_arclog switch [-D datadir]
pgx_arclog discard [-D datadir]
pgx_arclog register [-D datadir] ALFname
pgx_arclog inhibit [-D datadir] ALFname
pgx_arclog permit [-D datadir] ALFname
```

説明

アーカイブログファイルの作成、追加、削除、表示、退避、切り替え、登録、閉塞、復旧と閉塞解除などを行います。

詳細

createモード

アーカイブログファイルを作成します。

*ALFname*には、アーカイブログファイルのパス名を絶対パス名で指定します。

*ALFname*に指定できる長さは、英数字131バイトまでです。

*ALFsize*には、作成するアーカイブログファイルの大きさをメガバイト単位で指定します。数値の後ろに“M”を指定してください。

addモード

アーカイブログファイルを追加します。

*ALFname*には、追加するアーカイブログファイルのパス名を指定します。

dropモード

アーカイブログファイルを削除します。

*ALFname*には、削除するアーカイブログファイルのパス名を指定します。

viewモード

アーカイブログファイルの情報を表示します。

*ALFname*を指定した場合には、指定したアーカイブログファイルの情報を表示します。

*ALFname*を指定しない場合は、すべてのアーカイブログファイルの情報を表示します。

backupモード

アーカイブログファイルの内容を退避します。
*devicename*には、退避先となるファイルを作成する、一般ファイルのディレクトリ名を絶対パス名で指定します。
*ALFbackupname*には、アーカイブログファイルの退避先となるファイル名を指定します。
*devicename*に指定できる値の長さは、120バイトまでです。
*ALFbackupname*に指定できる値の長さは、120バイトまでです。

switchモード

アーカイブログファイルを切り替えます。

discardモード

退避動作待ちのアーカイブログファイルを破棄し、使用可能にします。

registerモード

作成済のアーカイブログファイルをログ管理ファイルに再登録します。
*ALFname*には、再登録を行う、作成済のアーカイブログファイルのパス名を絶対パスで指定します。

inhibitモード

アーカイブログファイルを閉塞します。
*ALFname*には、閉塞するアーカイブログファイルのパス名を指定します。

permitモード

アーカイブログファイルの復旧と閉塞解除を行います。
*ALFname*には、復旧するアーカイブログファイルのパス名を指定します。

オプション

pgx_arclogは以下のコマンドライン引数を受け付けます。

-D *datadir*

ファイルシステム内でのデータベースファイルの格納場所を指定します。
ディレクトリ名に指定できる長さは、100バイトまでです。
省略時は、環境変数 PGDATA が有効となります。

環境

PGDATA

データベースファイルの格納場所を指定します。-Dオプションを使用して上書きすることができます。

診断

0：正常終了
0以外：異常終了

注釈

createモード、dropモードおよびregisterモードは、PostgreSQL Plusが起動されていないときに、実行することができます。
inhibitモードおよびpermitモードは、PostgreSQL Plusが起動されているとき

に、実行することができます。
addモード、viewモード、backupモード、switchモードおよびdiscardモードは、PostgreSQL Plusが起動されているかどうかに関わらず、実行することができます。
このコマンドを実行する場合、インスタンス管理者用ユーザで実行してください。

関連項目

pgx_ctllog(1), pgx_tmplog(1)

1.2 pgx_ctllog

Name

pgx_ctllog - ログ管理ファイルの操作

Synopsis

```
pgx_ctllog create [-D datadir]
pgx_ctllog view [-D datadir]
pgx_ctllog inhibit [-D datadir]
           pgx_ctllog permit [-D datadir]
```

説明

ログ管理ファイルの作成、表示、閉塞、復旧と閉塞解除を行います。

詳細

createモード

ログ管理ファイルを作成します。

viewモード

ログ管理ファイルに関する情報を表示します。

inhibitモード

ログ管理ファイルを閉塞します。

permitモード

ログ管理ファイルの復旧と閉塞解除を行います。

オプション

pgx_ctllogは以下のコマンドライン引数を受け付けます。

-D *datadir*

ファイルシステム内でのデータベースファイルの格納場所を指定します。
ディレクトリ名に指定できる長さは、100バイトまでです。
省略時は、環境変数 PGDATA が有効となります。

環境

PGDATA

データベースファイルの格納場所を指定します。-Dオプションを使用して上書きすることができます。

診断

0：正常終了
0以外：異常終了

注釈

createモードは、PostgreSQL Plusが起動されていないときに、実行することができます。

inhibitモードおよびpermitモードは、PostgreSQL Plusが起動されているときに、実行することができます。
viewモードは、PostgreSQL Plusが起動されているかどうかに関わらず、実行することができます。
このコマンドを実行する場合、インスタンス管理者用ユーザで実行してください。

関連項目

pgx_arclog(1), pgx_tmplog(1)

1.3 pgx_tmplog

Name

pgx_tmplog - テンポラリログファイルの操作

Synopsis

```
pgx_tmplog create [-D datadir] [-c recoverylogsize] logindexpathname
        { BIlogpathname | -in } { AIlogpathname | -in | -bi }
          BIlogsize AIlogsize transactionentryno
pgx_tmplog alter [-D datadir] [-c recoverylogsize]
        { BIlogsize | - } { AIlogsize | - } [ transactionentryno ]
pgx_tmplog drop [-D datadir] [-e]
pgx_tmplog view [-D datadir]
pgx_tmplog register [-D datadir] logindexpathname
pgx_tmplog recreate [-D datadir] [-c recoverylogsize] logindexpathname
        { BIlogpathname | -in } { AIlogpathname | -in | -bi }
          BIlogsize AIlogsize transactionentryno
pgx_tmplog inhibit [-D datadir]
pgx_tmplog permit [-D datadir]
```

説明

テンポラリログファイルの作成、変更、削除、表示、登録、再作成、閉塞、復旧と閉塞解除などを行います。

テンポラリログファイル（ログインデックス域）を、以降、ログインデックス域と略します。

詳細

createモード

テンポラリログファイルを作成します。

*logindexpathname*には、テンポラリログファイルのログインデックス域のパス名を指定します。

*BIlogpathname*には、テンポラリログファイルのBIログ域のパス名を指定します。パス名の代わりに“-in”を指定するとログインデックス域と同一のUNIXファイルに割り当てます。

*AIlogpathname*には、テンポラリログファイルのAIログ域のパス名を指定します。パス名の代わりに“-in”を指定するとログインデックス域と同一のUNIXファイルに割り当てます。“-bi”を指定するとBIログ域と同一のUNIXファイルに割り当てます。

*BIlogsize*には、作成するテンポラリログファイルのBIログ域の大きさをメガバイト単位で指定します。数値の最後に“M”を指定してください。

*AIlogsize*には、作成するテンポラリログファイルのAIログ域の大きさをメガバイト単位で指定します。数値の最後に“M”を指定してください。

*transactionentryno*には、テンポラリログファイルで管理するトランザクショ

ン情報のエントリ数を指定します。同時実行するトランザクション数以上の値を指定する必要があります。
logindexpathname、*BIlogpathname*、*AIlogpathname*には、英数字131バイトまで指定でき、絶対パス名で指定します。ファイルシステム上のUNIXファイルを指定できます。

alterモード

テンポラリログファイルの属性を変更します。
*BIlogsize*には、変更後のBIログ域の大きさを指定します。“-”を指定するとBIログ域の大きさは変更されません。
*AIlogsize*には、変更後のAIログ域の大きさを指定します。“-”を指定するとAIログ域の大きさは変更されません。
*transactionentryno*には、変更後のトランザシクョン情報のエントリ数を指定します。
BIlogsize、*AIlogsize*、*transactionentryno*の指定内容に関しては、createモードの説明を参照してください。

dropモード

テンポラリログファイルを削除します。

viewモード

テンポラリログファイルの情報を表示します。

registerモード

作成済のテンポラリログファイルをログ管理ファイルに再登録します。
*logindexpathname*には、作成済のテンポラリログファイルに対するログインデックス域のパス名を絶対パスで指定します。

recreateモード

テンポラリログファイルを再作成します。指定する各パラメータに関しては、createモードの説明を参照してください。

inhibitモード

テンポラリログファイルを閉塞します。

permitモード

テンポラリログファイルの復旧と閉塞解除を行います。

オプション

pgx_tmplogは以下のコマンドライン引数を受け付けます。

-D *datadir*

ファイルシステム内でのデータベースファイルの格納場所を指定します。
ディレクトリ名に指定できる長さは、100バイトまでです。
省略時は、環境変数 PGDATA が有効となります。

-c *recoverylogsize*

サーバ起動時の復旧に使用するログ量をメガバイト単位で指定します。数値の後ろに“M”を指定してください。

-e

テンポラリログファイルを強制的に削除する場合に指定します。テンポラリログファイルが破壊された場合に使用します。

環境

PGDATA

データベースファイルの格納場所を指定します。-Dオプションを使用して上書きすることができます。

診断

0：正常終了
0以外：異常終了

注釈

createモード、alterモード、dropモード、registerモードおよびrecreateモードは、PostgreSQL PLusが起動されていないときに、実行することができます。
inhibitモードおよびpermitモードは、PostgreSQL Plusが起動されているときに、実行することができます。
viewモードは、PostgreSQL Plusが起動されているかどうかに関わらず、実行することができます。
このコマンドを実行する場合、インスタンス管理者用ユーザで実行してください。

関連項目

pgx_arclog(1), pgx_ctllog(1)

付録B サンプルスクリプト

ここでは、モニタリングおよびディザスタリカバリのサンプルスクリプトについて説明します。

B.1 モニタリングのサンプルスクリプト

ここでは、モニタリングを実行するスクリプトの例について説明します。サンプルスクリプトは、psqlを使用してシステムカタログを参照し、アクセス禁止状態を検出した場合、Eメールで異常を通知する処理を行うスクリプトの例です。

- スクリプトを動作させるための環境ファイル(config)
  アクセス禁止状態を検出するスクリプトを動作させるための環境ファイルです。

- アクセス禁止状態検出スクリプト(monitor.sh)
  psqlを使用してシステムカタログを参照し、アクセス禁止状態を検出するスクリプトです。

- メール送信スクリプト(sendMail.sh)
  アクセス禁止状態検出スクリプトでアクセス禁止状態を検出した場合、詳細情報をメールで送信するスクリプトです。

config

アクセス禁止状態を検出するスクリプトを動作させるための環境ファイルです。サンプルスクリプトを実行する環境に合わせて変更します。

```
#動作環境設定############################################################

#psqlの実行パス(必須)
PSQL_PATH="/usr/local/pgsqlplus/bin/psql"

# ユーザ名(必須)
PGUSER="postgres"

# アプリケーションとの通信のポート番号(必須)
PGPORT="5432"

#ログ用ディレクトリ(必須)
LOG_DIR="/var/tmp/monitor"
```

```
#ログファイル名(省略時:monitor.log)
LOG_FILE="monitor.log"

#ログの保存(省略時:0)
# ログに1回分の履歴を保存するか すべての履歴を保存するかを指定します
# 0 : 1回分の履歴を保存します
# 1 : すべての履歴を保存します
LOG_OPTION=0

#作業用ディレクトリ(必須):ディレクトリを作成し 終了時に削除します
TMP_DIR="/tmp/monitor"

#通知(省略時:1)
# 0:アクセス禁止状態が発生した場合 通知します
# 1:アクセス禁止状態が発生した場合
#    およびスクリプトで異常が発生した場合 通知します
# 2:通知しません
NOTICE_OPTION=1

#件名
# アクセス禁止状態を検出した場合のメールの件名
# (省略時:"Detected prohibited access in the PostgreSQL Plus resources. : データベース名")
ABNORMAL_MAIL_TITLE="Detected prohibited access in the PostgreSQL Plus resources. : ${PGDATABASE}"

# スクリプトで異常が発生した場合のメールの件名
# (省略時:"An error occurred in the monitoring of PostgreSQL Plus. : データベース名")
ERROR_MAIL_TITLE="An error occurred in the monitoring of PostgreSQL Plus. : ${PGDATABASE}"



############################################################
#通知形式
# 異常原因により 以下のような形式で異常情報をメールで通知します
#
#件名:
#ABNORMAL_MAIL_TITLEで指定した件名 (各異常原因ともに共通)
#
#本文:
#-データベース異常の場合の異常資源情報
#-----------------------------------------------------------#
# [異常発生]                                                #
# データベースの異常                                        #
# [異常資源]                                                #
# #データベーススペース                                     #・・・(1)異常資源名
# 原因,データベーススペース名,デバイス名                    #・・・(2)タイトル行
```

```
# MRV DBSP1,/PlusData/postgres/DBSP01.dat                  #・・・(3)異常情報
#    :                                                     #
# #シーケンス                                              #・・・(1)異常資源名
# 原因,スキーマ名.シーケンス名                             #・・・(2)タイトル行
# DER,public.sec_id_seq                                    #・・・(3)異常情報
#    :                                                     #
# #テーブル                                                #・・・(1)異常資源名
# 原因,スキーマ名.テーブル名,割付先テーブルスペース名      #・・・(2)タイトル行
# MRV,public.section_master,TBSP1                          #・・・(3)異常情報
#    :                                                     #
# #インデックス                                            #・・・(1)異常資源名
# 原因,スキーマ名,インデックス名,割付先テーブルスペース名  #・・・(2)タイトル行
# TIH,kanri.kanri_id_idx,TBSP2                             #・・・(3)異常情報
#    :                                                     #
#-----------------------------------------------------------#
#
#-バックアップ異常の場合の異常資源情報
#-----------------------------------------------------------#
# [異常発生]                                               #
# バックアップの異常                                       #
# [異常資源]                                               #
# #アーカイブログ                                          #・・・(1)異常資源名
# アーカイブパス名                                         #・・・(2)タイトル行
# /PlusBackup/postgres/arclog04.dat                        #・・・(3)異常情報
#    :                                                     #
#-----------------------------------------------------------#
#
#-サーバ全体異常の場合の異常資源情報
#-----------------------------------------------------------#
# [異常発生]                                               #
# サーバ全体の異常                                         #
# [異常資源]                                               #
# #データベーススペース                                    #・・・(1)異常資源名
# 原因,データベーススペース名,デバイス名                   #・・・(2)タイトル行
# MRV DBSP1,/PlusData/postgres/DBSP01.dat                  #・・・(3)異常情報
#    :                                                     #
# #シーケンス                                              #・・・(1)異常資源名
# 原因,スキーマ名.シーケンス名                             #・・・(2)タイトル行
# DER,public.sec_id_seq                                    #・・・(3)異常情報
#    :                                                     #
# #テーブル                                                #・・・(1)異常資源名
# 原因,スキーマ名.テーブル名,割付先テーブルスペース名      #・・・(2)タイトル行
# MRV,public.section_master,TBSP1                          #・・・(3)異常情報
#    :                                                     #
```

```
# #インデックス                                                  #・・・(1)異常資源名
# 原因,スキーマ名,インデックス名,割付先テーブルスペース名 #・・・(2)タイトル行
# TIH,kanri.kanri_id_idx,TBSP2                                  #・・・(3)異常情報
#    :                                                          #
# #アーカイブログ                                               #・・・(1)異常資源名
# アーカイブパス名                                              #・・・(2)タイトル行
# /PlusBackup/postgres/arclog04.dat                             #・・・(3)異常情報
#    :                                                          #
#-------------------------------------------------------------#
#
#[異常資源情報]
#異常情報の区切り(省略時:表示を省略します)
#異常情報と異常情報を区切るために使用((1)異常資源名,(3)異常情報で使用)
DELIM=","
#
#(1)異常資源名(省略時:表示を省略します)
#異常資源名を変更する場合 編集します
DBSPACES="#データベーススペース"
SEQUENCES="#シーケンス"
TABLE="#テーブル"
INDEX="#インデックス"
ARCHIVE_LOG="#アーカイブログ"
#
#(2)タイトル行(省略時:表示を省略します)
#異常情報のタイトル行を変更したい場合 編集します
#${DELIM}・・・異常情報の区切り
DBSPACES_TITLE="原因${DELIM}データベーススペース名${DELIM}デバイス名"
SEQUENCES_TITLE="原因${DELIM}スキーマ名.シーケンス名"
TABLES_TITLE="原因${DELIM}スキーマ名.テーブル名${DELIM}割付先テーブルスペース名"
INDEXES_TITLE="原因${DELIM}スキーマ名.インデックス名${DELIM}割付先テーブルスペース名"
ARCHIVE_LOG_TITLE="アーカイブパス名"
#
#"(3)異常情報"を取得するSQL文(必須)
#取得する異常情報を変更する場合 編集します
DBSPACES_SQL="SELECT cause,dbspacename,device FROM pgx_dbspaces WHERE status='INH' ORDER BY cause"
SEQUENCES_SQL="SELECT cause,schemaname||'.'||sequencename FROM pgx_sequences WHERE status='INH'
ORDER BY cause"
TABLES_SQL="SELECT cause,schemaname||'.'||tablename,tablespacename FROM pgx_tables WHERE status='INH'
ORDER BY cause"
INDEXES_SQL="SELECT cause,schemaname||'.'||indexname,tablespacename FROM pgx_indexes WHERE status='INH'
ORDER BY cause"
ARCHIVE_LOG_SQL="SELECT archivepath FROM pgx_archive_log WHERE status='inh' ORDER BY archivepath"

#####################################################################
```

変数名	指定値	変更
PSQL_PATH	psqlの実行パスを絶対パスで指定します。	○
PGUSER(注1)	データベースに接続するユーザを指定します。	○
PGPORT	アプリケーションとの通信のポート番号を指定します。	○
LOG_DIR	ログを出力するディレクトリを絶対パスで指定します。	○
LOG_FILE	ログを出力するファイル名を指定します。	△
LOG_OPTION	ログの保存方式を指定します。 0 ： 1回分の履歴を保存します 1 ： すべての履歴を保存します	△
TMP_DIR	スクリプトで使用する作業用のディレクトリを絶対パスで指定します。	○
NOTICE_OPTION(注2)	メールによるエラー通知を行う契機を指定します。 0 ： アクセス禁止状態が発生した場合、通知します 1 ： アクセス禁止状態が発生した場合およびスクリプトで異常が発生した場合、通知します 2 ： 通知しません	△
ABNORMAL_MAIL_TITLE	アクセス禁止状態を検出した場合のメールの件名を指定します。	△
ERROR_MAIL_TITLE	スクリプトで異常が発生した場合のメールの件名を指定します。	△
DELIM	異常情報の区切りを指定します。	△
DBSPACES	データベーススペースのアクセス禁止状態を検出した場合に、異常を検出した資源がデータベーススペースであることを示す文字列を指定します。	△
SEQUENCES	データベーススペースのアクセス禁止状態を検出した場合に、異常を検出した資源がシーケンスであることを示す文字列を指定します。	△
TABLE	データベーススペースのアクセス禁止状態を検出した場合に、異常を検出した資源がテーブルであることを示す文字列を指定します。	△
INDEX	データベーススペースのアクセス禁止状態を検出した場合に、異常を検出した資源がインデックスであることを示す文字列を指定します。	△
ARCHIVE_LOG	データベーススペースのアクセス禁止状態を検出した場合に、異常を検出した資源がアーカイブログであることを示す文字列を指定します。	△
DBSPACES_TITLE	pgx_dbspacesシステムカタログでアクセス禁止状態を検出した場合の詳細情報のタイトルを指定します。	△
SEQUENCES_TITLE	pgx_sequencesシステムカタログでアクセス禁止状態を検出した場合の詳細情報のタイトルを指定します。	△

変数名	指定値	変更
TABLES_TITLE	pgx_tablesシステムカタログでアクセス禁止状態を検出した場合の詳細情報のタイトルを指定します。	△
INDEXES_TITLE	pgx_indexesシステムカタログでアクセス禁止状態を検出した場合の詳細情報のタイトルを指定します。	△
ARCHIVE_LOG_TITLE	pgx_archive_logシステムカタログでアクセス禁止状態を検出した場合の詳細情報のタイトルを指定します。	△
DBSPACES_SQL	pgx_dbspacesシステムカタログを参照するselect文を指定します。	△
SEQUENCES_SQL	pgx_sequencesシステムカタログを参照するselect文を指定します。	△
TABLES_SQL	pgx_tablesシステムカタログを参照するselect文を指定します。	△
INDEXES_SQL	pgx_indexesシステムカタログを参照するselect文を指定します。	△
ARCHIVE_LOG_SQL	pgx_archive_logシステムカタログを参照するselect文を指定します。	△

○ ： 実行する環境に合わせて変更
△ ： 必要に応じて変更

注1)：データベースtemplate1にも接続可能なユーザを指定します。また、パスワードなしでデータベースに接続可能なユーザを指定します。
注2)：メールを送信できない環境の場合は、2を指定します。

monitor.sh

アクセス禁止状態を検出するスクリプトです。
アクセス禁止状態を検出するために、psqlを使用してシステムカタログを参照しています。

```
#!/bin/sh
#------------------------------------------------------------------------------#
#モニタリングサンプルツール
# 呼出        : monitor.sh〈データベース名〉
# パラメタ    :〈データベース名〉・・・ 監視対象のデータベース名(必須)
# 終了コード : 0 ・・・ 正常終了
#              1 ・・・ 正常終了(アクセス禁止状態発生)
#              2 ・・・ ツールのエラー終了
# 詳細        : アクセス禁止状態を検出し 異常資源とその原因をメールで通知します
#------------------------------------------------------------------------------#
BASE_DIR=`dirname $0`
cd ${BASE_DIR}
```

```
#リターンコード
RETURN_NORMAL=0
RETURN_DB_ABNORMAL=1
RETURN_COMMAND_ERROR=2

#--------------------------------------------------------------------------------#
#パラメタの取得
#--------------------------------------------------------------------------------#
PGDATABASE=$1
#パラメタが取得できなかった場合 エラーとします
if [ -z "${PGDATABASE}" ]; then
        echo "There is an error in the parameter."
        exit ${RETURN_COMMAND_ERROR}
fi

. ${BASE_DIR}/config

#メール送信ツール
SENDMAIL_TOOL="${BASE_DIR}/sendMail.sh"

MSG_BEGIN="Starts monitoring of PostgreSQL Plus."
MSG_INH_LOG_BEGIN="---Starts log output for prohibited access.------------------"
MSG_INH_LOG_END="---Stops log output for prohibited access.-------------------"
MSG_MAIL_ERROR="An error occurred in sending a mail."
MSG_FILE_ERROR="Failed to record additional information of the access-prohibited resources."
MSG_ER_MAIL_ERROR="An error occurred in sending an error mail."
MSG_END_NORMAL="Stops monitoring of PostgreSQL Plus successfully."
MSG_END_ABNORMAL="Stops monitoring of PostgreSQL Plus successfully. (Detected prohibitted access.)
MSG_END_ERROR="Stops monitoring of PostgreSQL Plus abnormally."

NORMAL=0
ERROR=1

#異常フラグ
DB_ABNORMAL_FLG=${NORMAL}
BKDIR_ABNORMAL_FLG=${NORMAL}

#ツールエラー発生フラグ
TOOL_FLG=${NORMAL}


#--------#
#準備処理#
#--------#
```

```
#ログディレクトリが存在しない場合 エラーとします
if [ ! -d "${LOG_DIR}" ]; then
        echo "LOG_DIR(${LOG_DIR}) doesn't exist."
        exit ${RETURN_COMMAND_ERROR}
fi

if [ -z "${LOG_FILE}" ]; then
        LOG_FILE="monitor.log"
fi

LOG_FILE=${LOG_DIR}/${LOG_FILE}

#作業用ディレクトリが存在しない場合 作成します
if [ ! -d "${TMP_DIR}" ]; then
        mkdir ${TMP_DIR}
        if [ $? -ne 0 ]; then
                echo "TMP_DIR(${TMP_DIR}) doesn't exist."
                exit ${RETURN_COMMAND_ERROR}
        fi
fi

#作業ディレクトリ配下に一時ファイルを作成します
SQL_TMP="${TMP_DIR}/sql.tmp"
DETAIL_TMP="${TMP_DIR}/detail.tmp"
MESSAGE_TMP="${TMP_DIR}/message.tmp"
if [ -f "${SQL_TMP}" ]; then
        rm -r ${SQL_TMP}
fi
if [ -f "${DETAIL_TMP}" ]; then
        rm -r ${DETAIL_TMP}
fi
if [ -f "${MESSAGE_TMP}" ]; then
        rm -r ${MESSAGE_TMP}
fi

if [ -z "${LOG_OPTION}" ]; then
        LOG_OPTION=0
fi

#ログファイルに1回分の履歴を保存する場合 ログファイルを新規作成します
if [ "${LOG_OPTION}" != "1" ]; then
        :>${LOG_FILE}
fi
```

```
 echo "${MSG_BEGIN}(`date """+%Y/%m/%d %H:%M:%S"""`)" >> ${LOG_FILE}

 #================================================================================#
 #終了処理[ subEnd ]
 # 呼出          : subEnd <status>
 # パラメタ      : <status> ・・・ ステータス
 # 詳細          : 作業用ディレクトリを削除して ステータスで終了します
 #================================================================================#
 subEnd() {
        if [ -d "${TMP_DIR}" ]; then
                rm -r ${TMP_DIR}
        fi

        if [ $1 -eq ${RETURN_NORMAL} ]; then
                echo "${MSG_END_NORMAL}(`date """+%Y/%m/%d %H:%M:%S"""`)" >> ${LOG_FILE}
        elif [ $1 -eq ${RETURN_DB_ABNORMAL} ]; then
                echo "${MSG_END_ABNORMAL}(`date """+%Y/%m/%d %H:%M:%S"""`)" >> ${LOG_FILE}
        elif [ $1 -eq ${RETURN_COMMAND_ERROR} ]; then
                echo "${MSG_END_ERROR}(`date """+%Y/%m/%d %H:%M:%S"""`)" >> ${LOG_FILE}
        fi
        exit $1
 }
 #================================================================================#
 #ツールエラー処理[ subToolErrorEnd ]
 # 呼出          : subToolErrorEnd <status>
 # パラメタ      : <status> ・・・ ステータス
 # 詳細          : エラー通知を行い 終了処理を呼び出します
 #================================================================================#
 subToolErrorEnd() {
        if [ -z "${ERROR_MAIL_TITLE}" ]; then
                ERROR_MAIL_TITLE="An error occurred in the monitoring of PostgreSQL Plus. : ${PGDATABASE}"
        fi

        if [ "${NOTICE_OPTION}" != "0" -a "${NOTICE_OPTION}" != "2" ]; then
                ${SENDMAIL_TOOL} -t "${ERROR_MAIL_TITLE}" -d ${TMP_DIR} >> ${LOG_FILE} 2>&1
                if [ $? -ne 0 ]; then
                        echo "${MSG_ER_MAIL_ERROR}" >> ${LOG_FILE}
                        subEnd ${RETURN_COMMAND_ERROR}
                fi
        fi
        subEnd $1
 }
 #--------------------------------------------------------------------------------#
 #1.アクセス禁止状態の検出
```

```
#-------------------------------------------------------------------------------#
#アクセス禁止状態の検出を行うシステムカタログ名一覧
SYSTEM_CATALOGS="PGX_DBSPACES PGX_SEQUENCES PGX_TABLES PGX_INDEXES PGX_ARCHIVE_LOG"

#システムカタログごとにアクセス禁止状態を検出します
for SYSTEM_CATALOG in ${SYSTEM_CATALOGS}
do
        case ${SYSTEM_CATALOG} in
                "PGX_DBSPACES") #データベーススペース
                        RESOURCES="${DBSPACES}"
                        TITLE_LINE="${DBSPACES_TITLE}"
                        SQL="${DBSPACES_SQL}" ;;
                "PGX_SEQUENCES") #シーケンス
                        RESOURCES="${SEQUENCES}"
                        TITLE_LINE="${SEQUENCES_TITLE}"
                        SQL="${SEQUENCES_SQL}" ;;
                "PGX_TABLES") #テーブル
                        RESOURCES="${TABLE}"
                        TITLE_LINE="${TABLES_TITLE}"
                        SQL="${TABLES_SQL}" ;;
                "PGX_INDEXES") #インデックス
                        RESOURCES="${INDEX}"
                        TITLE_LINE="${INDEXES_TITLE}"
                        SQL="${INDEXES_SQL}" ;;
                "PGX_ARCHIVE_LOG") #アーカイブログ
                        RESOURCES="${ARCHIVE_LOG}"
                        TITLE_LINE="${ARCHIVE_LOG_TITLE}"
                        SQL="${ARCHIVE_LOG_SQL}" ;;
        esac

        #SQLが指定されていない場合 フラグをエラーにして処理を続けます
        if [ -z "${SQL}" ]; then
                echo  "SQL is not specified. (${SYSTEM_CATALOG})" >> ${LOG_FILE}
                TOOL_FLG=${ERROR}
                continue
        fi

        #データベーススペースの場合 template1に接続して アクセス禁止状態を確認します
        if [ "${SYSTEM_CATALOG}" = "PGX_DBSPACES" ]; then
                PGDB="template1"
        else
                PGDB="${PGDATABASE}"
        fi
```

```
        ${PSQL_PATH} -U ${PGUSER} -c "${SQL}" -o ${SQL_TMP} -p ${PGPORT} -t -A -F "${DELIM}" ${PGDB} 1>/dev/null 2>>
${LOG_FILE}
        PSQL_END=$?
        #エラーが発生した場合 フラグをエラーにして処理を続けます
        if [ ${PSQL_END} -ne 0 ]; then
                echo "An error occurred in psql.(exit code: ${PSQL_END}) : ${SYSTEM_CATALOG}" >> ${LOG_FILE}
                TOOL_FLG=${ERROR}
        fi

        #アクセス禁止状態が発生していない場合 次のループ処理を行います
        if [ ! -s "${SQL_TMP}" ]; then
                continue
        fi

        if [ "${SYSTEM_CATALOG}" = "PGX_ARCHIVE_LOG" ]; then
                BKDIR_ABNORMAL_FLG=${ERROR}
        else
                DB_ABNORMAL_FLG=${ERROR}
        fi

        if [ -n "${RESOURCES}" ]; then
                echo "${RESOURCES}" >> ${DETAIL_TMP}
        fi
        if [ -n "${TITLE_LINE}" ]; then
                echo "${TITLE_LINE}" >> ${DETAIL_TMP}
        fi
        cat ${SQL_TMP} >> ${DETAIL_TMP}
        #エラーが発生した場合 フラグをエラーにして処理を続けます
        if [ $? -ne 0 ]; then
                echo "${MSG_FILE_ERROR} (${SQL_TMP})" >> ${LOG_FILE}
                TOOL_FLG=${ERROR}
        fi
done

#アクセス禁止状態が発生していない場合 正常終了します
if [ ${DB_ABNORMAL_FLG} -eq ${NORMAL}  -a  ${BKDIR_ABNORMAL_FLG} -eq ${NORMAL} ]; then
        #エラーが発生した場合 ツールエラーで終了します
        if [ ${TOOL_FLG} -eq ${ERROR} ]; then
                subToolErrorEnd ${RETURN_COMMAND_ERROR}
        fi
        #正常終了
        subEnd ${RETURN_NORMAL}
fi
```

```
#-------------------------------------------------------------------------------#
#2.メール通知
#-------------------------------------------------------------------------------#
if [ ${DB_ABNORMAL_FLG} -eq ${ERROR}  -a  ${BKDIR_ABNORMAL_FLG} -eq ${ERROR} ]; then
        ABNORMAL_CAUSE="サーバ全体の異常"
elif [ ${DB_ABNORMAL_FLG} -eq ${ERROR} ]; then
        ABNORMAL_CAUSE="データベースの異常"
elif [ ${BKDIR_ABNORMAL_FLG} -eq ${ERROR} ]; then
        ABNORMAL_CAUSE="バックアップの異常"
fi

#異常原因を作成します
#####################################################################
#異常原因より前に表示する内容を変更する場合 編集します

echo "[異常発生]" >> ${MESSAGE_TMP}

#####################################################################

#異常原因(必須)
echo "${ABNORMAL_CAUSE}" >> ${MESSAGE_TMP}

#####################################################################
#異常原因より後に表示する内容を変更する場合 編集します

echo "[異常資源]" >> ${MESSAGE_TMP}

#####################################################################

cat ${DETAIL_TMP} >> ${MESSAGE_TMP}
#エラーが発生した場合 フラグをエラーにして処理を続けます
if [ $? -ne 0 ]; then
        echo "${MSG_FILE_ERROR} (${DETAIL_TMP})" >> ${LOG_FILE}
        TOOL_FLG=${ERROR}
fi

#異常資源の情報をログに出力します
echo  "${MSG_INH_LOG_BEGIN}" >> ${LOG_FILE}
cat ${MESSAGE_TMP} >> ${LOG_FILE}
echo  "${MSG_INH_LOG_END}" >> ${LOG_FILE}

#-------------------------------------------------------------------------------#
#メールで通知します
#-------------------------------------------------------------------------------#
```

```
if [ "${NOTICE_OPTION}" != "2" ]; then
        if [ -z "${ABNORMAL_MAIL_TITLE}" ]; then
                ABNORMAL_MAIL_TITLE="Detected prohibited access in the PostgreSQL Plus resources. : ${PGDATABASE}"
        fi
        ${SENDMAIL_TOOL} -t "${ABNORMAL_MAIL_TITLE}" -m ${MESSAGE_TMP} -d ${TMP_DIR} >> ${LOG_FILE} 2>&1
        #エラーが発生した場合 フラグをエラーにして処理を続けます
        if [ $? -ne 0 ]; then
                echo "${MSG_MAIL_ERROR}" >> ${LOG_FILE}
                #ツールエラー処理で通知を行わないように設定します
                NOTICE_OPTION=2
                TOOL_FLG=${ERROR}
        fi
fi

#エラーが発生した場合 ツールエラーで終了します
if [ ${TOOL_FLG} -eq ${ERROR} ]; then
        subToolErrorEnd ${RETURN_COMMAND_ERROR}
fi

subEnd ${RETURN_DB_ABNORMAL}
```

sendMail.sh

メールを送信するスクリプトです。
メールの送信には、sendmailを使用しています。

```
#!/bin/sh
#-------------------------------------------------------------------------------#
#メール送信
# 呼出      : sendMail.sh -t <件名> -d <作業ディレクトリ> -m <本文ファイル>
# パラメタ  : <件名>              ・・・ メールの件名(必須)
#             <作業ディレクトリ>  ・・・ 存在する作業用ディレクトリ(必須)
#             <本文ファイル>      ・・・ メールの本文用ファイル名
# 終了コード: 0 ・・・ 正常終了
#             1 ・・・ エラー終了
# 詳細      : sendmailでメールを送信します
#             本文ファイルが指定されていない場合 件名のみで送信します
#             シェルの実行には 以下が必須です
#              -sendmail ・・・ メール管理プログラム
#             メールの本文に日本語を使用する場合 以下が必須です
#              -nkf ・・・ ネットワーク用漢字コード変換フィルタ
#-------------------------------------------------------------------------------#
```

```
##動作環境設定############################################################
#
#sendmailの実行パス(必須)
SENDMAIL="/usr/lib/sendmail"

#送信メールアドレス(必須)
FROM_ADDRESS=kanri@example.com

#受信メールアドレス(必須)
TO_ADDRESS=kanri@example.com

#メールの本文に日本語を使用する場合 以下が必須です
#nkf(ネットワーク用漢字コード変換フィルタ)の実行パス
NKF="/usr/bin/nkf"
#########################################################################

BASE_DIR=`dirname $0`
cd ${BASE_DIR}

#リターンコード
RETURN_NORMAL=0
RETURN_ERROR=1

#-------------------------------------------------------------------------------#
#パラメタ取得
#-------------------------------------------------------------------------------#
while getopts d:m:t: NAME
       do
       case ${NAME} in
                d) MAIL_DIR="$OPTARG";;
                m) MESSAGE_TMP="$OPTARG";;
                t) MAIL_TITLE="$OPTARG";;
                *) echo "Unexpected paramters are specified."
       esac
done

#件名が指定されていない場合 または、
#作業ディレクトリが指定していない/存在しない場合 エラーとします
if [ -z "${MAIL_TITLE}" -o ! -d "${MAIL_DIR}" ]; then
       echo "Parameters are not specified."
       exit ${RETURN_ERROR}
fi

#作業ディレクトリ配下に一時ファイルを作成します
```

```
MAIL_TMP="${MAIL_DIR}/mail.tmp"
if [ -f "${MAIL_TMP}" ]; then
	rm -r ${MAIL_TMP}
fi

#==============================================================================#
#終了処理[ subEnd ]
# 呼出		: subEnd <ステータス>
# パラメタ	:<ステータス> ・・・ 終了コード
# 詳細		: 一時ファイルを削除して ステータスで終了します
#==============================================================================#
subEnd() {
	#一時ファイルを削除します
	if [ -f "${MAIL_TMP}" ]; then
		rm -r ${MAIL_TMP}
	fi
	exit $1
}

#------------------------------------------------------------------------------#
#送信内容を作成します
#------------------------------------------------------------------------------#
#送信/受信アドレスがない場合 エラーとします
if [ -z "${FROM_ADDRESS}" -o -z "${TO_ADDRESS}" ]; then
	echo "Mail addresses to receive and send mails are not specified."
	subEnd ${RETURN_ERROR}
fi

#ヘッダを作成します
cat > ${MAIL_TMP} << EOF
From: ${FROM_ADDRESS}
To: ${TO_ADDRESS}
Subject: ${MAIL_TITLE}
EOF

###############################################################################
#メールのヘッダに文字コードを指定します
# メールの本文に含める文字コードによって cat～EOFまでを編集します
# 注意:cat～EOFの途中ではコメントが使用できません
# 以下は メールの本文に日本語を使用する場合の例です

cat >> ${MAIL_TMP} << EOF
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-2022-jp
```

```
Content-Transfer-Encording: 7bit
EOF

#########################################################################

#ヘッダと本文を区切る空行を追加します
echo "" >> ${MAIL_TMP}

#-------------------------------------------------------------------------------#
#本文ファイルからメールの本文を作成します
#-------------------------------------------------------------------------------#
if [ -f "${MESSAGE_TMP}" ]; then

        ######################################################################
        #本文ファイルを追加します
        #以下は メールの本文に日本語を使用する場合の例です

        ${NKF} -j ${MESSAGE_TMP} >> ${MAIL_TMP}
        #メールの本文の作成に失敗してもエラーとしません
        if [ $? -ne 0 ]; then
                echo "NKF command failed."
        fi

        ######################################################################

fi

#-------------------------------------------------------------------------------#
#メールを送信します
#-------------------------------------------------------------------------------#
${SENDMAIL} -ti < ${MAIL_TMP}
MAIL_END=$?
if [ ${MAIL_END} -ne 0 ]; then
        echo "Failed to send a mail. (exit code: ${MAIL_END})"
        subEnd ${RETURN_ERROR}
fi

subEnd ${RETURN_NORMAL}
```

変数名	指定値	変更
SENDMAIL	sendmailの実行パスを絶対パスで指定します。	○
FROM_ADDRESS	メールの送信者のアドレスを指定します。	○

変数名	指定値	変更
TO_ADDRESS	メールの受信者のアドレスを指定します。	○
NKF(注)	nkf(ネットワーク用漢字コード変換フィルタ)の実行パスを絶対パスで指定します。	○

○ ： 実行する環境に合わせて変更

注)：メールの本文に日本語を使用する場合、NKFが必要となります。

B.2　ディザスタリカバリのサンプルスクリプト

ここでは、“第4章 運用手順”で説明しているディザスタリカバリのバックアップ手順とリカバリ手順を実行するサンプルスクリプトについて説明します。

サンプルスクリプトでは、各種ディレクトリを作成します。インスタンス管理者用ユーザが書き込み可能な権限にしてください。

B.2.1　全体バックアップを実行する

全体バックアップを実行するスクリプトの例について説明します。
サンプルスクリプトを実行する場合、サンプルスクリプトを実行する環境に合わせて、以下の値を変更してください。

- インスタンスを格納するディレクトリ
  インスタンスを格納するディレクトリを絶対パスで指定します。
- バックアップディレクトリ
  バックアップディレクトリを絶対パスで指定します。
- 二次退避域のディレクトリ
  二次退避域のディレクトリを絶対パスで指定します。
- PostgreSQL Plusの実行パス
  PostgreSQL Plusの実行パスを絶対パスで指定します。

```
#!/bin/sh
# このスクリプトは、全体バックアップを行うシェルファイルです

# -- 環境設定
--------------------------------------------------------------------------
# それぞれの環境に合わせて変更してください

# インスタンスを格納するディレクトリを指定します
PGDATA=/PlusData/data1
```

```
# バックアップディレクトリを指定します
PGBKUP=/PlusBackup/data1

# 二次退避域のディレクトリを指定します
PGARC=/PlusArchive/data1

# PostgreSQL Plusの実行パスを指定します
PLUSPATH=/usr/local/pgsqlplus/bin

# エラーメッセージ出力関数 ----------------------------------------------------------
# 本スクリプトでエラーが発生した場合 exit code 1 で復帰します
error ()
{
      echo $tejun failed.
      echo `basename $0` did not complete.
      exit 1
}

# メイン処理開始--------------------------------------------------------------------
# インスタンス全体のバックアップを取得 (1)
tejun=full-copy_1
$PLUSPATH/pgx_dmpall -D $PGDATA || error

# 二次退避域のディレクトリの削除 (2)
if [ -d $PGARC ] ; then
      # 必要ならば, ここで過去のバックアップを削除する前に 過去のバックアップを退避します
      # 本スクリプトでは 削除します
      rm -rf $PGARC || error
fi

# 二次退避域のディレクトリの作成 (3)
mkdir -p $PGARC || error

# バックアップデータだけを二次退避域へ退避 (4)
tejun=full-copy_2
cp -rf $PGBKUP/bkup1 $PGARC || error

# 退避が完了した場合は exit code 0 で復帰します
echo `basename $0` completed.
exit 0
```

1. インスタンス全体のバックアップを取得

インスタンス全体のバックアップを取得しています。

2. 二次退避域のディレクトリの削除
   取得したバックアップを退避するために、過去のバックアップを削除しています。過去のバックアップを退避する必要がある場合、二次退避域のディレクトリを削除する前に退避してください。
3. 二次退避域のディレクトリの作成
   バックアップデータを退避するための二次退避域のディレクトリを作成しています。
4. バックアップデータだけを二次退避域へ退避
   バックアップデータだけを二次退避域へ退避しています。

B.2.2 差分バックアップを実行する

差分バックアップを実行するスクリプトの例について説明します。
サンプルスクリプトを実行する場合、サンプルスクリプトを実行する環境に合わせて、以下の値を変更してください。

- インスタンスを格納するディレクトリ
  インスタンスを格納するディレクトリを絶対パスで指定します。
- 二次退避域のディレクトリ
  二次退避域のディレクトリを絶対パスで指定します。
- PostgreSQL Plusの実行パス
  PostgreSQL Plusの実行パスを絶対パスで指定します。

```
#!/bin/sh
# このスクリプトは 差分バックアップを行うシェルファイルです

# -- 環境設定 ------------------------------------------------------------------------
# それぞれの環境に合わせて変更してください

# インスタンスを格納するディレクトリを指定します
PGDATA=/PlusData/data1

# バックアップの二次退避域のディレクトリを指定します
PGARC=/PlusArchive/data1

# PostgreSQL Plusの実行パスを指定します
PLUSPATH=/usr/local/pgsqlplus/bin

# エラーメッセージ出力関数 ---------------------------------------------------------
# 本スクリプトでエラーが発生した場合 exit code 1 で復帰します
error ()
{
```

```
        echo $tejun failed.
        echo `basename $0` did not complete.
        exit 1
}

# メイン処理開始 ------------------------------------------------------------

# 二次退避域にバックアップデータが存在していることを確認 (1)
tejun=diff-copy_0
if [ ! -d $PGARC/bkup1 ] ; then
        echo $PGARC/bkup1 does not exist.
        error
fi

# 復旧ポイントの確定(2)
tejun=diff-copy_1
s_msg=`$PLUSPATH/pgx_arclog switch -D $PGDATA 2>&1`
if [ $? -ne 0 ] ; then
        echo $s_msg
        error
fi

# アーカイブログファイルの状態確認 (3)
tejun=diff-copy_2
$PLUSPATH/pgx_arclog view -D $PGDATA | awk '$2 == "inh" {code=1} END {exit code}'
if [ $? -eq 1 ] ; then
        echo archive log status is inh.
        error
fi

# アーカイブログファイルを二次退避域へ退避 (4)
tejun=diff-copy_3
# アーカイブログファイルのステータスが switch または full となっているアーカイブログファイルを
退避します
$PLUSPATH/pgx_arclog view -D $PGDATA | awk '$2 == "switch" || $2 == "full" {print $2, $8}' | while
read status path
do
        # 二次退避域に退避されていないものを退避します
        if [ ! -f $PGARC/`basename $path` ] ; then
                cp -f $path $PGARC
                if [ $? -ne 0 ] ; then
                        rm $PGARC/`basename $path`
                        error
                fi
```

```
        fi
done

# 退避が完了した場合は exit code 0 で復帰します
echo `basename $0` completed.
exit 0
```

1. 二次退避域にバックアップデータが存在していることを確認
   差分バックアップは、全体バックアップが前提となるため、バックアップデータが存在していることを確認しています。
2. 復旧ポイントの確定
   復旧するポイントを確定しています。
3. アーカイブログファイルの状態確認
   アーカイブログファイルの状態を確認しています。
4. アーカイブログファイルを二次退避域へ退避
   退避する必要のあるアーカイブログファイルを二次退避域へ退避します。
   サンプルスクリプトでは、退避が必要な状態のアーカイブログファイルのうち、二次退避域に存在しないアーカイブログファイルだけを退避しています。

B.2.3 動作環境バックアップを実行する

動作環境バックアップを実行するスクリプトの例について説明します。
サンプルスクリプトを実行する場合、サンプルスクリプトを実行する環境に合わせて、以下の値を変更してください。

- インスタンスを格納するディレクトリ
  インスタンスを格納するディレクトリを絶対パスで指定します。
- バックアップディレクトリ
  バックアップディレクトリを絶対パスで指定します。
- 二次退避域のディレクトリ
  二次退避域のディレクトリを絶対パスで指定します。
- PostgreSQL Plusの実行パス
  PostgreSQL Plusの実行パスを絶対パスで指定します。

```
#!/bin/sh
# このスクリプトは、動作環境バックアップを行うシェルファイルです

# -- 環境設定 ------------------------------------------------------------
# それぞれの環境に合わせて変更してください

# インスタンスを格納するディレクトリを指定します
```

```
PGDATA=/PlusData/data1

# バックアップディレクトリを指定します
PGBKUP=/PlusBackup/data1

# バックアップの二次退避域のディレクトリを指定します
PGARC=/PlusArchive/data1

# PostgreSQL Plusの実行パスを指定します
PLUSPATH=/usr/local/pgsqlplus/bin

# 二次退避域のbkup1フォルダです 変更しないでください
ArcDir=$PGARC/bkup1

# エラーメッセージ出力関数 ------------------------------------------------------------
# 本スクリプトでエラーが発生した場合 exit code 1 で復帰します
error ()
{
       echo $tejun failed.
       echo `basename $0` did not complete.
       exit 1
}

# メイン処理開始-----------------------------------------------------------------------
# 二次退避域にバックアップデータが存在していることを確認 (1)
tejun=conf-copy_0
if [ ! -d $ArcDir ] ; then
       echo $ArcDir does not exist.
       error
fi

# 各種設定ファイルだけをバックアップ (2)
tejun=conf-copy_1
$PLUSPATH/pgx_dmpall -D $PGDATA -c || error


# 各種設定ファイルを二次退避域へ退避 (3)
tejun=conf-copy_2
for file in $PGBKUP/bkup1/* ; do
       case $file in
                $PGBKUP/bkup1/#*) ;;
                *) cp -f $file $ArcDir || error ;;
       esac
done
```

```
# 退避が完了した場合は exit code 0で復帰します
echo `basename $0` completed.
exit 0
```

1. 二次退避域にバックアップデータが存在していることを確認
   動作環境バックアップは、全体バックアップが前提となるため、バックアップデータが存在していることを確認しています。
2. 各種設定ファイルだけをバックアップ
   各種設定ファイルだけをバックアップしています。
3. 各種設定ファイルを二次退避域へ退避
   ファイル名の先頭文字が“#”でないファイルを、二次退避域に退避しています。

B.2.4 リカバリを実行する

リカバリを実行するスクリプトの例について説明します。
サンプルスクリプトを実行する場合、サンプルスクリプトを実行する環境に合わせて、以下の値を変更してください。

- インスタンスを格納するディレクトリ
  インスタンスを格納するディレクトリを絶対パスで指定します。
- バックアップディレクトリ
  バックアップディレクトリを絶対パスで指定します。
- データベーススペース
  データベーススペースを絶対パスで指定します。
- 二次退避域のディレクトリ
  二次退避域のディレクトリを絶対パスで指定します。
- 1個のアーカイブログファイルのサイズ
  1個のアーカイブログファイルのサイズを指定します。
- アーカイブログファイルの数
  アーカイブログファイルの数を指定します。
- PostgreSQL Plusの実行パス
  PostgreSQL Plusの実行パスを絶対パスで指定します。

```
#!/bin/sh
# このスクリプトは リカバリを行うシェルファイルです

# -- 環境設定 ------------------------------------------------------------------
# それぞれの環境に合わせて変更してください

# インスタンスを格納するディレクトリを指定します
PGDATA=/PlusData/data1
```

```
# バックアップディレクトリを指定します
PGBKUP=/PlusBackup/data1

# データベーススペースを指定します
DBSPACE=/PlusDbspace/data1

# バックアップの二次退避域のディレクトリを指定します
PGARC=/PlusArchive/data1

# 1個のアーカイブログファイルのサイズを指定します
ARCSIZE=50M

# アーカイブログファイルの数を指定します
ARCNUM=10

# PostgreSQL Plusの実行パスを指定します
PLUSPATH=/usr/local/pgsqlplus/bin

# エラーメッセージ出力関数 ------------------------------------------------------
# 本スクリプトでエラーが発生した場合 exit code 1 で復帰します
error ()
{
        echo $tejun failed.
        echo `basename $0` did not complete.
        exit 1
}

# ディレクトリの再作成 (1) - a)
tejun=Preparation-for-recovery_1

# 再作成するディレクトリが存在するかどうか 確認します
if [ -d $PGDATA ] ;then
        echo $PGDATA exist.
        error
fi
if [ -d $PGBKUP ] ;then
        echo $PGBKUP exist.
        error
fi
for i in $DBSPACE
do
        if
                [ -d $i ] ; then
```

```
                echo $i exist.
                error
        fi
done

# ディレクトリを作成します
mkdir -p $PGDATA || error
mkdir -p $PGBKUP || error
for i in $DBSPACE
do
        mkdir -p $i || error
done

# ディレクトリの権限の設定 (1) - b)

tejun=Preparation-for-recovery_2
chmod 700 $PGDATA $PGBKUP || error
for i in $DBSPACE
do
        chmod 700 $i || error
done

# バックアップデータのリストア (2) - a)
tejun=Restore-of-the-backup-resources_1
cp -rf $PGARC/bkup1 $PGBKUP || error

# バックアップデータを格納するbkup1ディレクトリの登録 (2) - b)
tejun=Restore-of-the-backup-resources_2
ln -s bkup1 $PGBKUP/backup || error

# アーカイブログファイルのリストア (2) - c)
tejun=Restore-of-the-backup-resources_3
arc=0
for file in $PGARC/*
do
        case $file in
                $PGARC/arclog*) cp -f $file $PGBKUP && arc=1 || error ;;
        esac
done

# 各種設定ファイルの準備 (3) - a)
tejun=Recovery-of-the-backup-directory_1
cp -f $PGBKUP/bkup1/edmf.conf $PGDATA/ || error
cp -f $PGBKUP/bkup1/edmfpool.conf $PGDATA/ || error
```

```
cp -f $PGBKUP/bkup1/postgresql.conf $PGDATA/ || error

# インスタンスを格納するディレクトリにバックアップディレクトリを登録（3）- b）
tejun=Recovery-of-the-backup-directory_2
ln -s $PGBKUP $PGDATA/backupdir || error

# インスタンスを格納するディレクトリにログ情報を登録）（3）- c）
tejun=Recovery-of-the-backup-directory_3
$PLUSPATH/pgx_ctllog create -D $PGDATA || error

# テンポラリログの作成（3）- d）
tejun=Recovery-of-the-backup-directory_4
mkdir $PGDATA/pgx_tlog || error
chmod 700 $PGDATA/pgx_tlog || error
$PLUSPATH/pgx_tmplog create -D $PGDATA -c 64 $PGDATA/pgx_tlog/TLF.log -in -in 50M 20M 64 || error

# アーカイブログファイルの作成（3）- e）
tejun=Recovery-of-the-backup-directory_5
arclog_create=0
count=1
while [ $count -le $ARCNUM ]
do
        number=`printf "%03d" $count`
        if [ ! -f $PGBKUP/arclog$number ] ; then
                if [ $arclog_create -eq 0 ] ; then
        $PLUSPATH/pgx_arclog create -D $PGDATA $PGBKUP/arclog$number $ARCSIZE && arclog_create=1
|| error
                else
                        $PLUSPATH/pgx_arclog add -D $PGDATA $PGBKUP/arclog$number || error
                fi
        fi
        count=`expr $count + 1`
done

# インスタンスの復旧（4）
tejun=Recovery-of-the-instance-storage-directory
if [ $arc -eq 1 ] ; then
        $PLUSPATH/pgx_rcvall -D $PGDATA -B $PGBKUP || error
else
        $PLUSPATH/pgx_rcvall -D $PGDATA -B $PGBKUP -p || error
fi

# 退避が完了した場合は exit code 0で復帰します
        echo `basename $0` completed.
```

```
exit 0
```

1. リカバリの前準備
    a) ディレクトリの再作成
    データベースを復旧するために必要なディレクトリを作成しています。サンプルスクリプトでは、データベースを復旧するために必要なディレクトリがすでに存在する場合、エラーとしています。
    b) ディレクトリの権限の設定
    作成したディレクトリを、インスタンス管理者用ユーザだけが書き込み可能な権限にしています。なお、サービスを起動するユーザとして、ローカルシステムアカウントを指定していた場合は、インスタンスを格納するディレクトリ、およびバックアップディレクトリの権限を、インスタンス管理者用ユーザに加え、サービス(SYSTEMグループ)もフルコントロールとなるように変更してください。

2. バックアップ資源のリストア
    a) バックアップデータのリストア
    二次退避域から、バックアップデータをリストアしています。
    b) バックアップデータを格納するbkup1ディレクトリの登録
    バックアップデータを格納するbkup1ディレクトリの登録をしています。
    c) アーカイブログファイルのリストア
    差分バックアップを行っている場合、二次退避域からアーカイブログファイルをリストアしています。

3. バックアップディレクトリの復旧
    a) 各種設定ファイルの準備
    バックアップディレクトリを復旧するために、各種設定ファイルをインスタンスを格納するディレクトリ配下にコピーしています。
    b) インスタンスを格納するディレクトリにバックアップディレクトリを登録
    インスタンスを格納するディレクトリにバックアップディレクトリを登録しています。
    c) インスタンスを格納するディレクトリにログ情報を登録
    インスタンスを格納するディレクトリにログ情報を登録しています。
    d) テンポラリログの作成
    テンポラリログを作成しています。

e)　　アーカイブログファイルの作成
バックアップディレクトリに存在しないアーカイブログファイルを作成しています。

4. インスタンスの復旧
a)　　差分バックアップを使用してリカバリする場合
バックアップデータとアーカイブログファイルを使用してインスタンスを復旧しています。
b)　　差分バックアップを使用しないでリカバリする場合
バックアップ時点への復旧になりますので、pgx_rcvallコマンドに-pオプションを指定して実行しています。