Rocket® software

スケールアウトパフォーマンス/可用性クラスターの利用

Version.2
Last updated: 2026 年 6 月

Rocket software

目次

各機能の詳細に関しては、製品マニュアルページからご利用になるバージョンを選択後、内容をご確認ください。
https://www.amc.rocketsoftware.co.jp/support/manuals.asp

1. はじめに

企業経営の主要な業務を支える基幹システムは、日々変化するマーケットのニーズに対応できる柔軟な環境や迅速にリリースできるプロセスの確立に加え、システム停止によるビジネスへの影響を回避するために、安定した稼働を強く求められます。

基幹システムの高可用性を実現する方法の1つとしてクラスタリングソフトを使用したシステムの冗長化が挙げられますが、これは物理的障害などが発生した場合に、アクティブサーバーとスタンバイサーバーを自動的に切り替え、システムの停止を回避するものです。

加えて、実行環境のスペックを増強する“スケールアップ”、実行環境を複数用意して処理の負荷を分散させる“スケールアウト”は、想定外の負荷による大幅な処理の遅延や、その結果としてのシステム停止を未然に防ぐ対策として、コンテナのオーケストレーションサービスやクラウドにおいても広く利用されています。

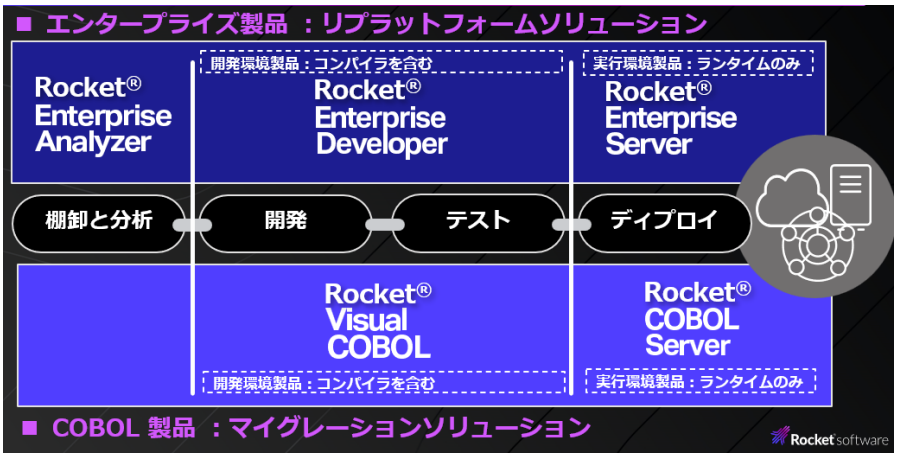
さらに、基幹システムに関連するすべてのソフトウェアが個別に高可用性を意識した機能を持ち合わせていれば、幾重にも対策を施すことができ、安定したシステムの提供が可能になります。

Enterprise Developer/Enterprise Server はスケールアウトパフォーマンス/可用性クラスター機能を備えており、複数の Enterprise Server インスタンスを単一システムイメージで運用、管理することができます。この構成を取り入れることにより、JCL 処理は負荷分散され、単一障害点を最小限に抑えることができます。

本文書では、Enterprise Developer/Enterprise Server 製品を使用したスケールアウトパフォーマンス/可用性クラスターを実現する方法について解説します。

2. モダナイゼーション支援製品

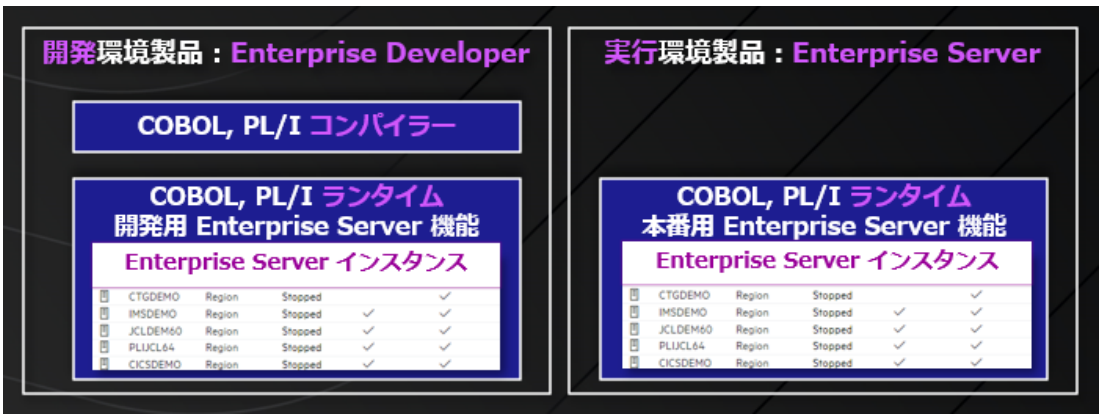
弊社のモダナイゼーション支援製品は、COBOL 製品群とエンタープライズ製品群の2つに分類されています。本文書ではエンタープライズ製品群に含まれる開発環境製品の Enterprise Developer と、実行環境製品の Enterprise Server に焦点を当てて解説します。



3. 開発環境製品と実行環境製品の違い

開発環境製品の Enterprise Developer はコンパイラとランタイムを含み、開発用マシンにインストールする製品です。業界標準 IDE の Eclipse、Visual Studio、Visual Studio Code を利用した開発作業や、開発用の実行環境機能により、開発工程における単体テストや JCL の実行、IDE を利用したデバッグなど、効率的なテストを行うことができます。

一方、実行環境製品の Enterprise Server はランタイムのみを含み、本番用マシンにインストールする製品です。実行単位となる Enterprise Server インスタンスは、例えば JCL を対象としたバッチ用、IMS、CICS オンライン用、Web サービス用など、運用用途に合わせて複数作成することができます。



4. スケールアウトパフォーマンス/可用性クラスターの目的と効果

1） 目的

基幹システムの停止はビジネスに甚大な影響を与え、企業の信頼喪失につながることから、常に安定した稼働を強く求められています。オープン環境においてもサーバーの物理的な障害、ネットワークの切断、ソフトウェアによる障害など様々な問題に備える必要があり、複数の視点から目的に応じたツールや回避策を施すことになります。

このようなニーズから、Enterprise Developer と Enterprise Server 製品には、スケールアウトパフォーマンス/可用性クラスター（以降 PAC と称す）機能が含まれており、これを利用することによりシステムの高可用性を実現します。

2） 効果

製品の PAC 機能を利用すると、次に挙げるようなシステムの高可用性を実現できます。

①負荷分散

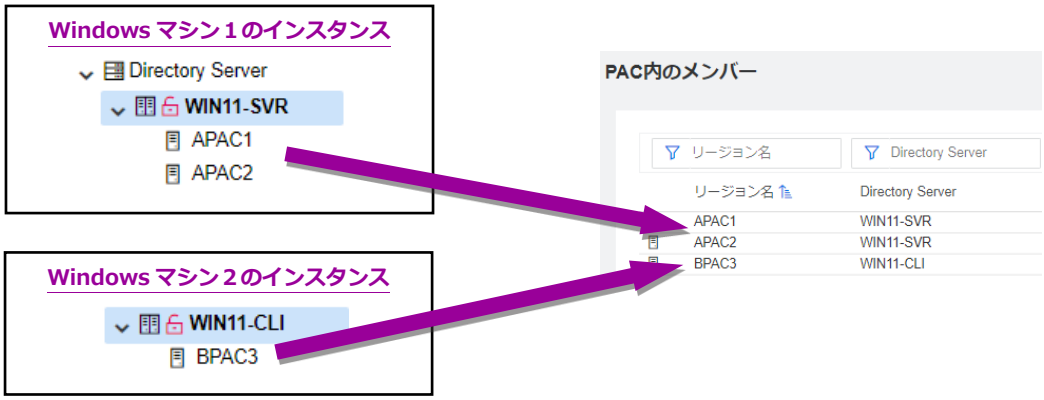
複数の Enterprise Server インスタンスを PAC 構成に参加させ、単一システムのイメージで処理の負荷分散を実現します。PAC 構成に登録された Enterprise Server インスタンスを PAC メンバーと呼びます。

例えば JCL を使用したバッチ処理では、自動的に PAC メンバーの空いているイニシエータへジョブが割り振られ、効率的に処理を実行することができます。

②単一障害点の回避

稼働させる OS や製品バージョン、稼働ビット数が一致すれば、IP アドレスの異なる複数のマシンに存在する Enterprise Server インスタンスを PAC メンバーに加えることができ、ハードウェアやネットワークに起因する単一障害点を最小限に抑えることができます。

2つの Windows 環境から PAC メンバーに指定した例）



③ファイルの共有と制御

PAC メンバー間で JES のカタログファイル、スプールファイル、CICS のリソース定義ファイル、処理に使用する VSAM などのデータファイルが共有でき、排他制御が行われます。
これらのファイルを Database File Handler（以降 DBFH と称す）機能を利用して、RDBMS へ格納することもでき、データの可用性を高めることもできます。

④ロードモジュールの共有

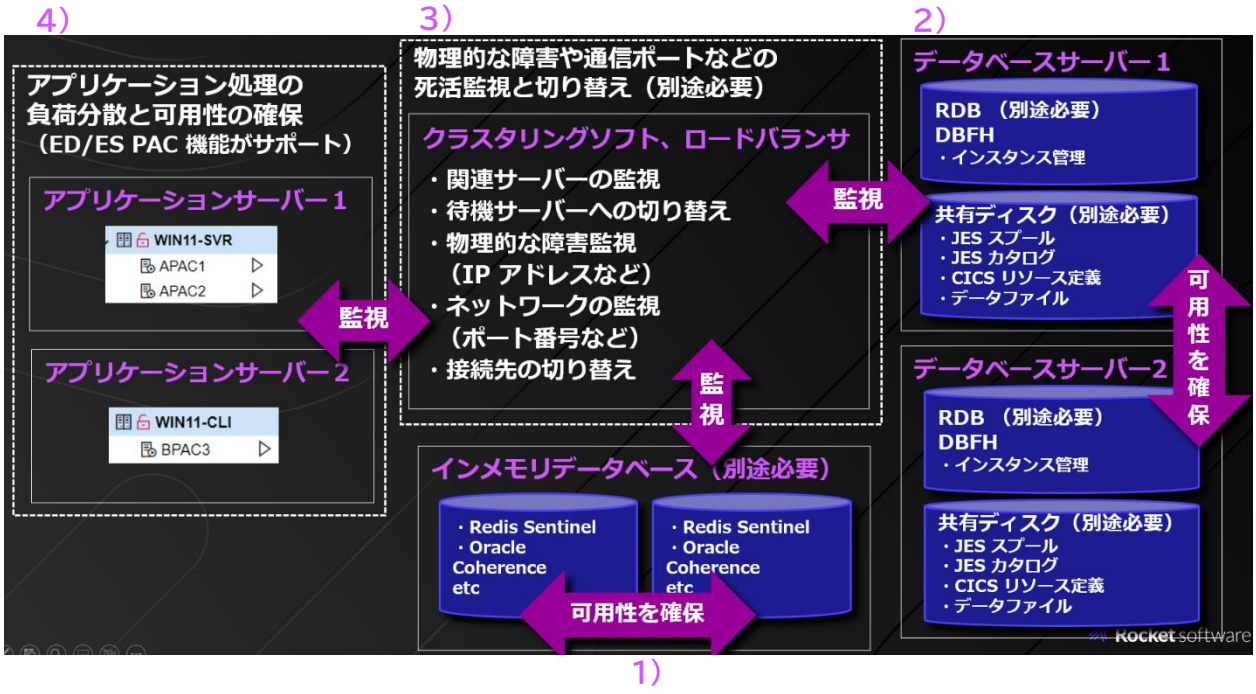
PAC メンバー間で実行可能ファイルであるロードモジュールを共有することができます。例えば1つの Enterprise Server インスタンスに対して CICS NEWCOPY を行った場合には、指定したプログラムが使用中でないことを前提に新しいモジュールがロードされ、すべての PAC メンバーへ伝播されます。

5. PAC を取り巻く環境

PAC 機能では、PAC メンバーの同期と管理のために必要となる構成要素があります。加えて、必要に応じて準備する製品以外のツールについてご説明します。

PAC 構成の例）

- RDBMS とインメモリデータベースを別途用意し、可用性を確保するように配置
- １つのアプリケーションサーバーから共通管理画面 Enterprise Server Common Web Administration（以降ESCWAと称す）を使用して、PACとスケールアウトレポジトリ（以降SORと称す）を構築
- アプリケーションサーバー1と2の Enterprise Server インスタンスを PAC メンバーに登録



1） インメモリデータベース

PAC メンバー間の同期を実現するために**インメモリデータベースに SOR を構築する必要**があります。PAC 構成に紐づけられた SOR を PAC スケールアウトレポジトリ（以降 PSOR と称す）と呼びます。

本番環境や本番相当のテスト環境では Redis Cluster、Redis Sentinel や Oracle Coherence の高可用性機能の導入を計画し、PAC 構成全体の高可用性を確保することを強くお勧めします。

2） RDBMS

PAC メンバーを管理するために、**RDBMS に DBFH を構築する必要**があります。サポートされている RDBMS を製品マニュアルでご確認後、別途ご準備ください。
PAC メンバーからのアクセスは、接続文字列または ODBC 経由となります。

RDBMS の機能を利用したスタンバイ環境との同期などを計画し、PAC 構成全体の高可用性を確保することを強くお勧めします。

3） ロードバランサ、クラスタリングソフト

基幹システムの停止を防ぐためには、サーバーのダウンやネットワーク障害などに備えてスタンバイ環境と同期とる、接続先を切り替えるなど複数の対策を施す必要があり、これらを実現する場合は別途ロードバランサやクラスタリングソフトをご準備ください。

一般的にクラスタリングソフトは、物理的な障害や OS レベルの障害によりサーバー自体がダウンした場合にスタンバイ環境へ切り替えるもの、ロードバランサは、処理の負荷分散とポートなどの死活監視、障害時の接続先切り替えを目的に導入するもので、エンタープライズ製品の PAC 機能は、業務処理の負荷分散を提供しています。

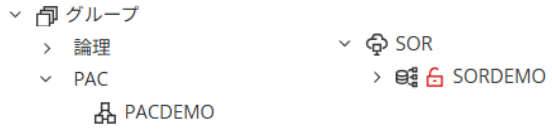
PAC 構成では、PAC メンバー間でファイルを共有できることから、ロードバランサが障害を検知した場合は、稼働している PAC メンバーに接続先を切り替えて業務処理を続行することもできます。

4） PAC、SOR、PSOR

PAC を構成するために、1つのマシンの ESCWA を使用して PAC や SOR、PSOR を登録しますが、例えばマシン1の ESCWA からこれらを登録し、マシン1がダウンした場合でも、他マシンの PAC メンバーが起動中であれば PAC 機能は有効であり、処理の負荷分散は継続されます。

また、他のマシンから PAC メンバーとして指定された Enterprise Server インスタンスは、PAC メンバーとして起動されます。

ローカルマシンの ESCWA で PAC、SOR を登録した例）



PAC メンバーとして起動されたことがコンソールログに記録）

```
CASSI1435I    00000 Other module(s) were synchronized from PAC PACDEMO CasLOADS.
CASSI1435I    00000 CICS module(s) were synchronized from PAC PACDEMO CasLOADS.
CASSI1435I    00000 JES module(s) were synchronized from PAC PACDEMO CasLOADS.
```

PAC のソフトウェア要件は、ご利用になるバージョンの製品マニュアルを開き、「ここからはじめよう ＞ 製品情報 ＞ Enterprise Developer のインストール ＞ 環境別の Readme」に含まれる「パフォーマンス・アベイラビリティ クラスター(PACs)のソフトウェア要件」欄をご参照ください。

6. PAC のコンポーネント

PAC 構成には下記の項目を目的として、必須コンポーネントと任意に構成できるオプションコンポーネントがあります。

1） Enterprise Server インスタンスの可用性

PAC 構成には Enterprise Server インスタンスの指定が必須です。何かの障害により1つのインスタンスが停止した際にも可用性を保持するために、3つ以上のインスタンスを PAC メンバーにすることを推奨しています。また、メンバーの同期に必要な PSOR を PAC 構成に対して必ず1つ指定します。

2） システムデータの可用性

PAC メンバー間におけるシステムデータの共有と管理のために、DBFH を利用したリージョンデータベースとリージョン間データベースの構成が必須です。

3） データファイルの可用性

データの可用性を高めるために、JES のカタログファイルや CICS のリソース定義ファイル、データファイルなどを DBFH が構築されている RDBMS へ格納し、管理することができます。ただし、書込み頻度が多い場合は、処理スピードに影響することがあります。この構成は必須ではありません。

4） システムのセキュリティ

PSOR や SOR にアクセスするクライアントに対して、認証を要求することができます。
この構成は必須ではありません。

5） SOR の負荷分散

CICS TD、TS キューのデータを別の SOR に格納することができます。これにより特定データのアクセスに対する負荷を分散することができますが、この構成は必須ではありません。

7. PAC 利用の前提条件

PAC 機能を利用するにあたり、下記の前提条件をご確認ください。

1） OS とサードパーティ製品

サポートされる OS、RDBMS、ODBC のバージョンは製品バージョンによって異なりますので、ご利用になるバージョンの製品マニュアルを開き、「ディプロイ ＞ 構成および管理 ＞ Enterprise Server の構成および管理 ＞ スケールアウト パフォーマンス/可用性クラスター ＞ 前提条件」をご確認ください。

例）

Windows

データベース	ODBC ドライバー
Microsoft SQL Server 2008 (Express Edition を含む) R2 以降	ODBC 13.1 for SQL Server
PostgreSQL 10.*x* (Amazon Aurora Postgres を含む) 以降	psqlodbc 11
DB2 10.5 以降	なし
Oracle 19c	なし

2） サポートされる構成と制限事項

PAC 構成と制限事項は製品バージョンによって異なりますので、ご利用になるバージョンの製品マニュアルを開き、「ディプロイ ＞ 構成および管理 ＞ Enterprise Server の構成および管理 ＞ スケールアウト パフォーマンス/可用性クラスター ＞ サポートされる構成」をご確認ください。

3） PAC メンバーの条件

PAC 構成に参加する PAC メンバーは以下の条件を満たす必要があります。

- Enterprise Server インスタンスが存在する OS のバージョンが同じであること。
- Enterprise Server インスタンスが存在するエンタープライズ製品のバージョンが同じであること。
- Enterprise Server インスタンスの稼働ビット数が同じであること。

8. デモ環境の概要

これから解説するデモでは Windows 環境を使用します。詳細は以下の通りです。

1） OS

Windows 11 Pro

2） プロセッサ

13th Gen Intel(R) Core(TM) i7-13800H (2.92 GHz) (2 プロセッサ)

3） システムの種類

64 ビット オペレーティング システム、x64 ベース プロセッサ

4） 各環境の名前解決

IP アドレスは hosts ファイルにて名前解決しています。

ファイルパスの例）C:¥Windows¥System32¥drivers¥etc¥hosts
設定値）
<ESCWA を運用するマシンの IP アドレス> WIN11-SVR

5） ユーザー権限

アクセスや書き込み権限のあるユーザーを使用しています。

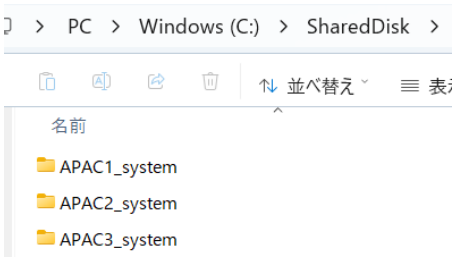
6） Micro Focus Directory Server（以降は EDS と称す）サービスの開始設定

異なる Windows 環境の EDS へアクセスして Enterprise Server を管理する場合は、これを許可するよう、参照される側の EDS 設定ファイルを変更します。Windows サービスから EDS サービスを停止後に変更し、EDS サービスを開始してください。念のため、オリジナルファイルはバックアップしてください。

ファイルパス）C:¥ProgramData¥Micro Focus¥Enterprise Developer¥mfdsacfg.xml
更新前の設定値）<bind_address>127.0.0.1</bind_address>
更新後の設定値）<bind_address>0.0.0.0</bind_address>

7） 共有フォルダ

すべての PAC メンバーが使用する共有フォルダを WIN11-SVR に作成しています。



8） インメモリデータベース

Redis をダウンロードして、WIN11-SVR 上にインストール後、redis-server.exe を実行しています。

9） DBFH

WIN11-SVR 上の PostgreSQL に DBFH を構築しています。

製品の DBFH 機能では、サポートしている RDBMS 別にスクリプトを提供しています。任意のデータベース名などを引数として環境構築用のスクリプトを生成し、これを実行することでデータベースやデータストアを作成することができます。

詳しくは、ご利用になるバージョンの製品マニュアルを開き、「ディプロイ ＞ 構成および管理 ＞ Enterprise Server の構成および管理 ＞ スケールアウト パフォーマンス/可用性クラスター ＞ PAC の構成 ＞ PAC の必須コンポーネントの構成 ＞ データベース接続の構成」をご参照ください。

10) XA スイッチモジュール

アプリケーションで使用するデータを DBFH に格納する場合は、XA スイッチモジュールをビルドして使用します。詳しくは、ご利用になるバージョンの製品マニュアルを開き、「ディプロイ ＞ 構成および管理 ＞ Enterprise Server の構成および管理 ＞ サーバー インスタンス環境の構成 ＞ XA 準拠リソース（XAR）の使用 ＞ RM スイッチ モジュール ＞ RM スイッチ モジュールのビルド」をご参照ください。

11) ODBC

DBFH へは、ODBC を設定して接続しています。

12) 稼働ビット数

プログラム、Enterprise Server インスタンス共に 64 ビット稼働を指定しています。

13) データ文字コード

製品では EBCDIC もしくは ASCII/SJIS 文字コードを指定できますが、デモ環境では ASCII/SJIS を使用しています。

14) ESCWA の表示文字コード

ESCWA の JES 機能で日本語が含まれるスプールやデータを表示する場合は、前もって製品インストールパスに利用する CCSID を展開し、MFACCCGI_CHARSET 環境変数を指定する必要があります。詳しい設定方法は、製品マニュアルの「Getting Started ＞ メインフレーム チュートリアル ＞ JCL チュートリアル」をご参照ください。

15) 使用した製品バージョン

Enterprise Developer 11.0J for Eclipse

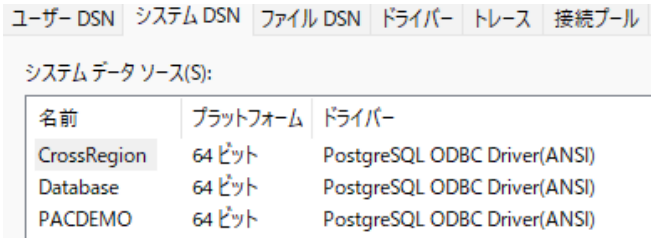
補足）Enterprise Server 11.0J と同等の開発用実行環境を含んでいます。

9. PAC の構成手順

PAC 構成に必要な要素やコンポーネントをご理解いただいた上で、ここからは設定する手順について具体的に解説します。

1） ODBC の作成

DBFH へ接続するために 64 ビットの ODBC を作成します。ODBC を設定せずとも DBFH 設定ファイルの文字列により接続は可能です。



CrossRegion：リージョン間（Enterprise Server インスタンス間）データベース用の ODBC

Database：データベース接続用の ODBC

PACDEMO：リージョンデータベース用の ODBC

2） DBFH 構成ファイルの作成

DBFH 環境へ接続するために MFDBFH.cfg という名前の構成ファイルを用意します。

環境情報）

データベース名：postgres

アクセス権限を持つユーザー：postgres

ユーザーのパスワード：password

内容例）

```
<?xml version="1.0" encoding="utf-8"?>
<datastores>
  <server name="WIN11-SVR" type="postgresql" access="odbc">
    <dsn name="Database" type="database" dbname="postgres" userid="postgres" password="password"/>
    <dsn name="PACDEMO" type="region.cas" region="PACDEMO" feature="all" userid="postgres" password="password"/>
    <dsn name="CrossRegion" type="crossregion.cas" userid="postgres" password="password"/>
  </server>
</datastores>
```

3行目：DBFH が存在する環境とその種類、アクセス方法を指定しています。

5行目：リージョンデータベースの指定となり、後述で作成する PAC 名を指定します。

6行目：リージョン間データベースの指定です。

配置場所は任意ですが、デモ環境ではこの構成ファイルを WIN11-SVR の共有フォルダである C:¥SharedDisk¥DBFH に配置しています。

このファイルはすべての PAC メンバーが参照します。

3） インメモリデータベースの開始

ESCWA から SOR を作成する前に、WIN11-SVR のコマンドプロンプトから、Redis を開始します。

コマンド例）

```
C:¥Windows¥System32>"C:¥Program Files¥Redis¥redis-server.exe"
```

異なるマシンからのアクセスが必要な場合は、末尾に“ --bind 0.0.0.0”を指定します。

デフォルトの 6379 ポートで Redis が開始されます。

4） Enterprise Server インスタンスの設定

ESCWA から PAC メンバーとなる Enterprise Server インスタンスの設定を行います。デモで使用するインスタンス名は以下の3つです。

WIN11-SVR マシンに存在するインスタンス）

APAC1

APAC2

APAC3

① 構成情報

各インスタンスに必要な環境変数を設定します。

追加設定

構成情報

```
[ES-Environment]
SHARED=c:\SharedDisk
MFACCCGI_CHARSET=Shift_JIS
MFCODESET=939
ES_DB_SERVER=WIN11-SVR
MFDBFH_CONFIG=$SHARED\DBFH\MFDBFH.cfg
```

・SHARED

共有フォルダのパスを環境変数として指定しています。

異なるマシンからのアクセスはネットワーク経由となるため、パスの先頭に¥¥¥¥を指定します。

・MFACCCGI_CHARSET

JES 機能において日本語を表示するため、Shift_JIS を指定します。

・MFCODESET

CICS 機能において使用する文字コードの 939 を指定しています。

・ES_DB_SERVER

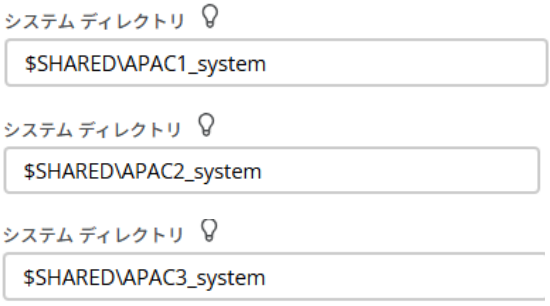
DBFH が構築されている RDBMS の存在する IP アドレスを指定します。

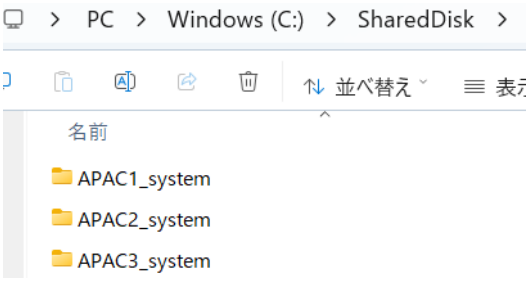
・MFDBFH_CONFIG

前述で作成した MFDBFH.cfg ファイルの完全パスを指定します。

② システムディレクトリ

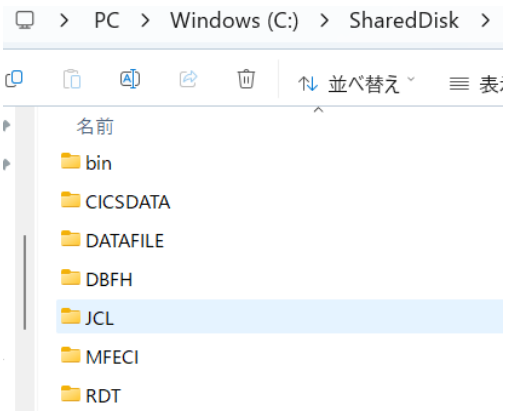
インスタンスごとに出力されるコンソールログなどを格納するシステムディレクトリは個別に設定する必要があります。デモ環境では共有フォルダにそれぞれのフォルダを用意しています。





③ CICS の構成

共有フォルダに格納されているリソース定義ファイル、実行モジュール、データファイル、マップファイルを共有するため、PAC メンバーに同じ値を設定しています。SIT についても同じ値を入力します。



④ JES の構成

共有フォルダに格納されている実行モジュールを共有するため、PAC メンバーの JES プログラムパスには同じ値を設定しています。

各 PAC メンバーには A から K クラスを実行できるイニシエータを1つ設定しています。
PAC 構成は1つのシステムイメージで稼働するため、このグループ内に同じクラスを実行するイニシエータが3つあることになります。並列実行により負荷分散が実現されますが、逆に並列実行を防ぐ場合は PAC メンバー内で該当クラスのイニシエータが単一になるよう設定します。

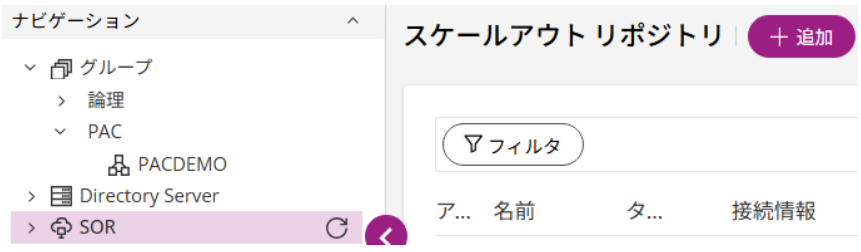
PAC 構成内においても、ジョブ名が重複している並列実行は制御され、デフォルト設定では先行ジョブが終了するまで後行ジョブは待機します。

⑤ XA リソースの登録

アプリケーションからデータベースに ODBC 接続する必要がある場合には、PAC メンバーの XA リソースの構成に前述でビルドした XA スイッチモジュールと ODBC 名を指定して登録します。

5） SOR の作成

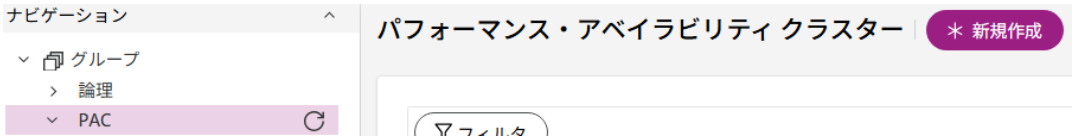
WIN11-SVR の ESCWA から SOR を作成します。左側メニューから SOR を選択し、右側ペインの追加ボタンをクリックします。



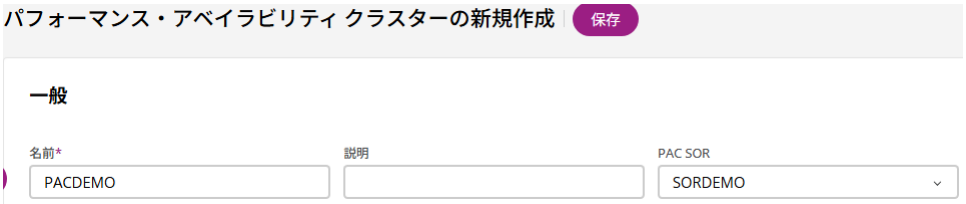
名前は任意ですが、デモ環境では SORDEMO を、タイプには Redis を、接続パスには前述で開始した Redis が存在するマシンの IP アドレスとポート番号を指定します。

6） PAC の作成

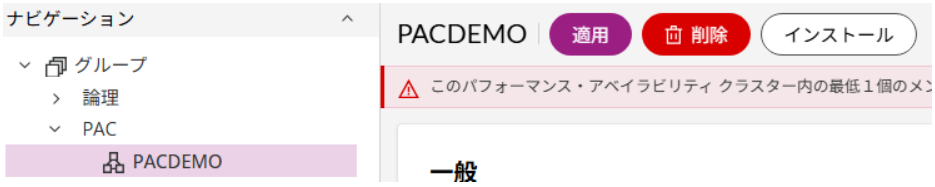
WIN11-SVR の ESCWA から PAC を作成します。左側メニューから PAC を選択し、右側ペインの追加ボタンをクリックします。



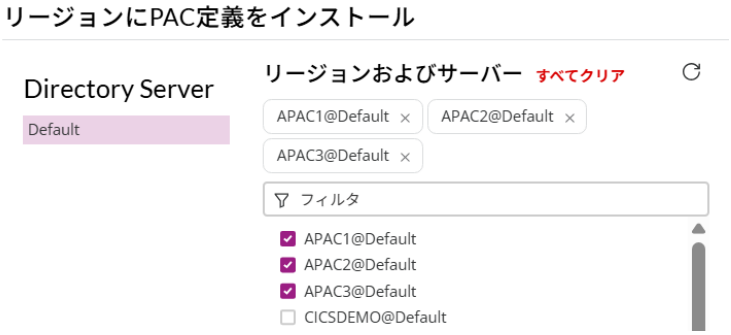
名前は任意ですが、デモ環境では PACDEMO を、PAC SOR には前述で作成した SORDEMO を選択します。



左側メニューから、追加した PACDEMO を選択し、右側ペインのインストールボタンをクリックします。

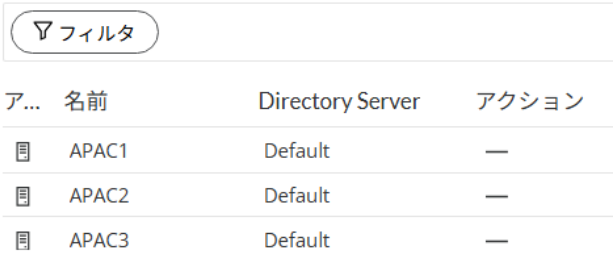


インストールウィンドウで、Enterprise Server インスタンスを PAC メンバーとして指定します。



3つのインスタンスをメンバーとした PAC 構成が作成されます。

PAC内のメンバー



10. PAC メンバーの開始

PAC メンバーを開始する具体的な手順について説明します。

1） PAC の初期化

必要に応じて、すべての PAC メンバーが停止されていることを確認後、caspac コマンドを使用して PAC の初期化を行います。これにより PSOR からすべての情報が削除されます。

Enterprise Developer コマンドプロンプトから、定義済みの PAC 名と前述で起動済みの Redis が存在するマシンの IP アドレスと接続ポート番号を指定します。

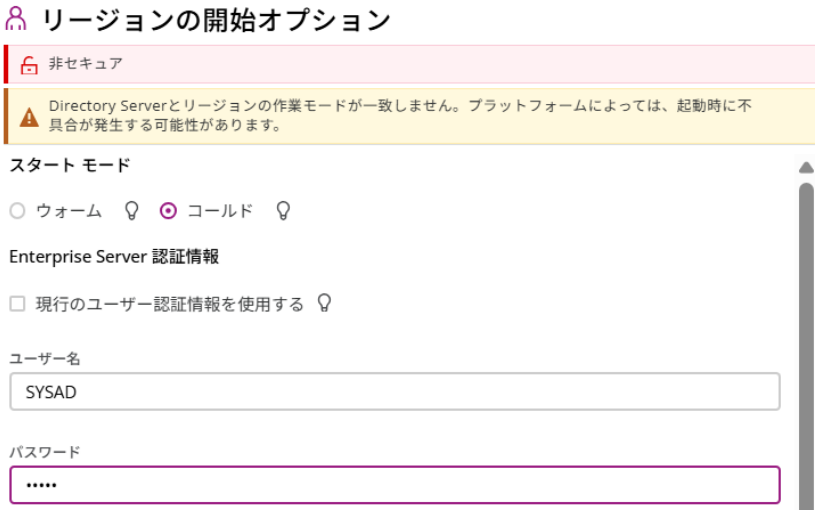
コマンド例）caspac -aInitPac=PACDEMO -sredis,WIN11-SVR:6379

```
C:\Users\tarot\Documents>caspac -aInitPac=PACDEMO -sredis,WIN11-SVR:6379
Connected to Redis server at WIN11-SVR port 06379
Using PACName PACDEMO
PAC PACDEMO initialisation return code     0

Highest return code:     0
```

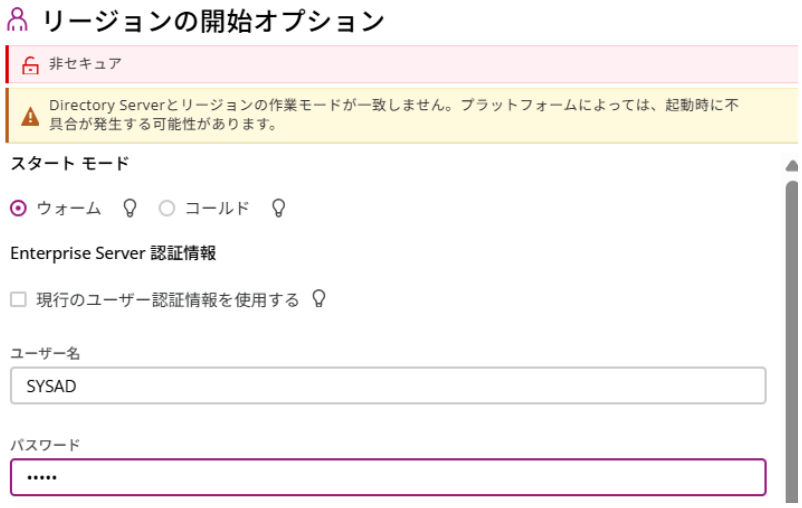
2） 最初に開始する PAC メンバー

関連する SOR の既存データをすべて削除するために、最初に開始する PAC メンバーにはコールドを指定して、開始権限のあるユーザーで開始します。デモ環境では APAC1 が該当します。このコールド指定は必須ではありません。



3） 2つめ以降に開始する PAC メンバー

2つ目以降の PAC メンバーは、関連する SOR の既存データを保持するために、ウォームを指定して開始権限のあるユーザーで開始します。デモ環境では APAC2、APAC3 が該当します。



正常に開始されない場合は、システムディレクトリにある console.log ファイルをテキストエディターで表示し、エラーの内容を確認します。

正常に開始された場合は、ESCWA から APAC1のコンソールログを表示し、他の2つのインスタンスと同期していることを確認します。

```
CASMG0001I  APAC2(002)#CASSI1432I Joining PAC PACDEMO. 14:25:28#
CASMG0001I  APAC2(002)#CASSI1435I 00000 Other module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:29#
CASMG0001I  APAC2(002)#CASSI1435I 00000 CICS module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:29#
CASMG0001I  APAC2(002)#CASSI1435I 00000 JES module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:29#
CASMG0001I  APAC3(003)#CASSI1432I Joining PAC PACDEMO. 14:25:37#
CASMG0001I  APAC3(003)#CASSI1435I 00000 Other module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:39#
CASMG0001I  APAC3(003)#CASSI1435I 00000 CICS module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:39#
CASMG0001I  APAC3(003)#CASSI1435I 00000 JES module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:39#
```

11. JCL の実行

3つの JCL を実行して、負荷分散とファイルの排他制御が行われているかを確認します。

1） SLEEPLCK.JCL

ジョブ名） SLEEPLCK

実行クラス）A

使用ファイル）

VSAM.KSDS1 ファイルを OUTPUT で OPEN

実行プログラム内容）

ファイルを OPEN 後、60 秒スリープし、

ソート後のデータを VSAM.KSDS1 ファイルへ出力

```
//SLEEPLCK JOB CLASS=A,MSGCLASS=A
//*************************************
//DEFVSAM1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DELETE VSAM.KSDS1 PURGE
  SET LASTCC=0
  DEFINE CLUSTER (NAME(VSAM.KSDS1)) -
   DATA -
   (RECORDS(10) FREESPACE(20 10) KEYS(5 0) -
   RECORDSIZE(71 71) ) -
   INDEX (RECORDS(50 50))
/*
//SORTSTEP EXEC SORTD
//SORT1.SORTIN DD *
00009Shiki Masaoka       5-5,Dogo Onsen,Matsuyama-shi,Ehime-ken    1870
00001Soseki Natsume      1-1,Koishikawa,Bunkyo-ku,Tokyo-to         1886
00007Ogai Mori           3-1,Rintaro-cho,Tsuwano-shi,Shimane-ken   1886
00002Ryotaro Shiba       2-3,Sonezaki,Kita-ku,Osaka-shi,Osaka-fu   1900
00006Jirocho Shimizu     6-6,Jiro-cho,Shimizu-shi,Shizuoka-ken     1800
/*
//SORT1.SORTOUT DD DSN=&&JINJIDAT,DISP=(NEW,PASS),
// SPACE=(800,(10,10)),DCB=(RECFM=FB,LRECL=71,DSORG=PS),UNIT=SYSDA
//SORT1.SYSIN DD *
  SORT FIELDS=(1,5,CH,A)

//APPL1 EXEC PGM=SLEEPLCK
//SYSOUT DD SYSOUT=*
//KSDSFILE DD DSN=VSAM.KSDS1,DISP=SHR
//INDD DD DSN=&&JINJIDAT,DISP=(OLD,DELETE)
//PRINTER DD SYSOUT=*
```

サブミットする Enterprise Server インスタンス）

APAC1:ポート番号 58884

2） NOWAIT.JCL

ジョブ名） NOWAIT

実行クラス）A

使用ファイル）

SAM.KSDS2 ファイルを OUTPUT で OPEN

実行プログラム内容）

ソート後のデータを VSAM.KSDS2 ファイルへ出力

サブミットする Enterprise Server インスタンス）

APAC1:ポート番号 58884

```
//NOWAIT JOB CLASS=A,MSGCLASS=A
//************************************
//DEFVSAM1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DELETE VSAM.KSDS2 PURGE
  SET LASTCC=0
  DEFINE CLUSTER (NAME(VSAM.KSDS2)) -
   DATA -
   (RECORDS(10) FREESPACE(20 10) KEYS(5 0) -
   RECORDSIZE(71 71) ) -
   INDEX (RECORDS(50 50))
/*
//SORTSTEP EXEC SORTD
//SORT1.SORTIN DD *
00009Shiki Masaoka       5-5,Dogo Onsen,Matsuyama-shi,Ehime-ken    1870
00001Soseki Natsume      1-1,Koishikawa,Bunkyo-ku,Tokyo-to         1886
/*
//SORT1.SORTOUT DD DSN=&&JINJIDAT,DISP=(NEW,PASS),
// SPACE=(800,(10,10)),DCB=(RECFM=FB,LRECL=71,DSORG=PS),UNIT=SYSDA
//SORT1.SYSIN DD *
  SORT FIELDS=(1,5,CH,A)

//APPL1 EXEC PGM=NOWAIT
//SYSOUT DD SYSOUT=*
//KSDSFILE DD DSN=VSAM.KSDS2,DISP=SHR
//INDD DD DSN=&&JINJIDAT,DISP=(OLD,DELETE)
//PRINTER DD SYSOUT=*
```

3） WAIT.JCL

ジョブ名） WAIT

実行クラス）C

使用ファイル）

VSAM.KSDS1 ファイルを I-O で OPEN

実行プログラム内容）

ソート後のデータを SLEEPLCK.JCL と同じ

VSAM.KSDS1 ファイルへ出力

サブミットする Enterprise Server インスタンス）

APAC3:ポート番号 59260

```
//WAIT JOB CLASS=C,MSGCLASS=C
//************************************
//SORTSTEP EXEC SORTD
//SORT1.SORTIN DD *
00005Eiji Yoshikawa      9-3,Miyamotomura,Mimasaka-gun,Okayama-ken 1920
00004Osamu Dazai         2-6,Tsugaru,Tsugaru-gun,Aomori-ken        1911
00008Ryoma Sakamoto      1-1,Harimayabashi,Kochi-shi,Kochi-ken     1820
00010Yukichi Fukuzawa    8-8,Keio-cho,Nakatsu-shi,Oita-ken         1835
00003Hideyo Noguchi      5-1,Inawashiro,Aizu-shi,Fukushima-ken     1911
/*
//SORT1.SORTOUT DD DSN=&&JINJIDAT,DISP=(NEW,PASS),
// SPACE=(800,(10,10)),DCB=(RECFM=FB,LRECL=71,DSORG=PS),UNIT=SYSDA
//SORT1.SYSIN DD *
  SORT FIELDS=(1,5,CH,A)

//APPL1 EXEC PGM=WAIT
//SYSOUT DD SYSOUT=*
//KSDSFILE DD DSN=VSAM.KSDS1,DISP=SHR
//INDD DD DSN=&&JINJIDAT,DISP=(OLD,DELETE)
//PRINTER DD SYSOUT=*
```

4） 実行

前述の JCL を順番に実行します。これらの JCL は共有フォルダに配置してあり、インスタンスの Web Service J2EE ポートへ実行権限のあるユーザーでサブミットします。

コマンド例）

```
cassub -jC:¥SharedDisk¥JCL¥SLEEPLCK.JCL -stcp:WIN11-SVR:58884 /uSYSAD /pSYSAD
cassub -jC:¥SharedDisk¥JCL¥NOWAIT.JCL -stcp:WIN11-SVR:58884 /uSYSAD /pSYSAD
cassub -jC:¥SharedDisk¥JCL¥WAIT.JCL -stcp:WIN11-SVR:59260 /uSYSAD /pSYSAD
```

```
C:\Users\tarot\Documents>cassub -jC:\SharedDisk\JCL\SLEEPLCK.JCL -stcp:WIN11-SVR:58884 /uSYSAD /pSYSAD
JCLCM0187I J0001000 SLEEPLCK JOB  SUBMITTED (JOBNAME=SLEEPLCK,JOBNUM=0001000) 14:45:16
JCLCM0172E J0001000 SLEEPLCK JOB  FAILED  - JCL ERROR 14:45:16
JES000050E JCL parsing error: Job: 0001000 RC =    8 14:45:16
Processed "C:\SharedDisk\JCL\SLEEPLCK.JCL"

C:\Users\tarot\Documents>cassub -jC:\SharedDisk\JCL\NOWAIT.JCL -stcp:WIN11-SVR:58884 /uSYSAD /pSYSAD
JCLCM0187I J0001001 NOWAIT   JOB  SUBMITTED (JOBNAME=NOWAIT,JOBNUM=0001001) 14:45:28
JCLCM0172E J0001001 NOWAIT   JOB  FAILED  - JCL ERROR 14:45:28
JES000050E JCL parsing error: Job: 0001001 RC =    8 14:45:28
Processed "C:\SharedDisk\JCL\NOWAIT.JCL"

C:\Users\tarot\Documents>cassub -jC:\SharedDisk\JCL\WAIT.JCL -stcp:WIN11-SVR:59260 /uSYSAD /pSYSAD
JCLCM0187I J0001002 WAIT     JOB  SUBMITTED (JOBNAME=WAIT,JOBNUM=0001002) 14:45:36
JCLCM0172E J0001002 WAIT     JOB  FAILED  - JCL ERROR 14:45:36
JES000050E JCL parsing error: Job: 0001002 RC =    8 14:45:36
Processed "C:\SharedDisk\JCL\WAIT.JCL"
```

5） 実行記録の確認

各 PAC メンバーのコンソールログを表示して実行記録を確認します。

APAC1 のコンソールログ）

① SLEEPLCK.JCL を受け付け、60 秒のスリープ後に正常終了しています。

② NOWAIT.JCL を受け付けていますが、実行記録がありません。SLEEPLCK ジョブがイニシエータを占有しているため、他のインスタンスで実行されているか後述で確認します。

	JCLCM0187I	J0001010 SLEEPLCK JOB SUBMITTED (JOBNAME=SLEEPLCK,JOBNUM=0001010)
	JCLCM0180I	J0001010 SLEEPLCK Job ready for execution.
	JES000004I	J0001010 SLEEPLCK JOB DISPATCHED
	JCLCM0188I	J0001010 SLEEPLCK JOB STARTED
②	JCLCM0187I	J0001011 NOWAIT JOB SUBMITTED (JOBNAME=NOWAIT,JOBNUM=0001011)
	JCLCM0180I	J0001011 NOWAIT Job ready for execution.
①	JCLCM0182I	J0001010 SLEEPLCK JOB ENDED - COND CODE 0000

APAC2のコンソールログ）

今回のサブミットでは使用されていません。

CASMG0001I	APAC3(003)#CASSI1432I Joining PAC PACDEMO. 14:25:37#
CASMG0001I	APAC3(003)#CASSI1435I 00000 Other module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:39#
CASMG0001I	APAC3(003)#CASSI1435I 00000 CICS module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:39#
CASMG0001I	APAC3(003)#CASSI1435I 00000 JES module(s) were synchronized from PAC PACDEMO CasLOADS. 14:25:39#

APAC3のコンソールログ）

NOWAIT.JCL は APAC1 のイニシエータが占有されているため、APAC3 で実行されています。

JES000004I	J0001011 NOWAIT JOB DISPATCHED
JCLCM0188I	J0001011 NOWAIT JOB STARTED
CASMG0001I	** START NOWAIT **
CASMG0001I	** END OF NOWAIT **
JCLCM0182I	J0001011 NOWAIT JOB ENDED - COND CODE 0000

WAIT.JCL は SLEEPLCK.JCL がファイルを開放するのを待って、APAC3 で実行されています。

JCLCM0187I	J0001012 WAIT JOB SUBMITTED (JOBNAME=WAIT,JOBNUM=0001012)
JCLCM0180I	J0001012 WAIT Job ready for execution.
JES000004I	J0001012 WAIT JOB DISPATCHED
JCLCM0188I	J0001012 WAIT JOB STARTED
MVSXR0091I	J0001012 WAIT Waiting for shared use of dataset "VSAM.KSDS1".
MVSXR0093I	J0001012 WAIT Required datasets have been acquired. Execution resumed.
CASMG0001I	** START WAIT **
CASMG0001I	** END OF WAIT **
JCLCM0182I	J0001012 WAIT JOB ENDED - COND CODE 0000

JES 機能のスプールを ESCWA から表示し、時間経過から結果を確認します。

ア...	名前	ジョブID	クラス	ユーザー	条件コード	サブミット		開始		完了	
	SLEEPLCK	J0001010	A	SYSAD	0000	①	15:56:57:57	①	15:56:57:67	②	15:57:57:88
	NOWAIT	J0001011	A	SYSAD	0000	②	15:57:01:01	②	15:57:01:07	①	15:57:01:20
	WAIT	J0001012	C	SYSAD	0000	③	15:57:03:83	③	15:57:03:87	③	15:57:58:06

3つの JCL はサブミット順に受け付けられ、開始されています。
ジョブが最初に完了したのは NOWAIT、次に SLEEPLCK、最後に WAIT となり意図した結果となりました。

この結果から PAC メンバー内における負荷分散とファイルの排他制御が行われ、正常に終了したことがわかります。

12. CICS の実行

TN3270 エミュレータから CICS 処理を実行します。

1） CICS トランザクションの実行

まずは、APAC1 の3270リスナーポート9005へアクセスし、ACCT トランザクションを実行して、一部のデータを修正します。

修正前のデータ）

```
ACCOUNT FILE: RECORD CHANGE

ACCOUNT NO: 11111       SURNAME:    JANICE
                        FIRST:      JOHN            MI: Y  TITLE: MR
TELEPHONE: 01688 1234   ADDRESS:    23 ROSE DRIVE
                                    EASTON

OTHERS WHO MAY CHARGE:
```

修正後のデータ）

```
ACCOUNT FILE: RECORD DISPLAY

ACCOUNT NO: 11111       SURNAME:    BALL
                        FIRST:      JOHN            MI: Y  TITLE: MR
TELEPHONE: 01688 1234   ADDRESS:    23 ROSE DRIVE
                                    EASTON

OTHERS WHO MAY CHARGE:
```

次に、APAC2 の 9010 ポートへアクセスしてログオンし、ACCT トランザクションを実行します。
修正したデータを確認します。

同じデータの表示結果）

```
ACCOUNT FILE: RECORD DISPLAY

ACCOUNT NO: 11111       SURNAME:    BALL
                        FIRST:      JOHN            MI: Y  TITLE: MR
TELEPHONE: 01688 1234   ADDRESS:    23 ROSE DRIVE
                                    EASTON

OTHERS WHO MAY CHARGE:
```

修正後のデータが確認できました。
この結果から、CICS リソース定義ファイル、FCT で指定されたデータファイルが PAC メンバーで共有されていることがわかります。

13. おわりに

企業の中枢を担う基幹システムは、「物理的障害の回避」、「ネットワークの安定性確保」、「業務処理停止の回避」など、多角的な視点でメインフレームと同等の堅牢性をオープン環境で実現する必要に迫られています。

オープン環境では堅牢性を確保するための様々なツールが提供されており、目的に応じて過不足のない機能を持ったツールを自由に選択することができます。また、ツールの種類によっては、移行後の運用コストダウンも期待することができます。

Enterprise Developer と Enterprise Server 製品は、アプリケーションの視点から堅牢性をサポートする機能がデフォルトで搭載されており、これによる効果は前述の通りです。

基幹システムの移行に向けて、弊社のモダナイゼーション支援製品と共に PAC 機能の導入をご検討いただければ幸いです。