構成マージを使用した InterSystems IRIS の自動構成

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

構成マージを使用した InterSystems IRIS の自動構成
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

図一覧

テーブル一覧

構成マージを使用した InterSystems IRIS の自動構成

このドキュメントでは、構成マージを使用して InterSystems IRIS を導入または再構成する方法を説明します。

1 構成マージとは

構成マージ機能により、InterSystems IRIS® インスタンスの構成に対して、1 回の操作で任意の数の変更を加えることができます。これを使用するには、導入時または後の任意の時点で、宣言型の構成マージ・ファイルに加える変更を記録し、そのファイルをインスタンスに適用するだけです。簡単に使用できる構成マージにより、同じコンテナ・イメージまたはキットからさまざまな構成の複数のインスタンスを自動的に導入でき、同時に複数のインスタンスを再構成し、自動化されたクラスタの再構成またはその他の複数インスタンスの導入を可能にします。構成マージは、サポートされている任意の UNIX®、または Linux プラットフォーム (Linux クラウド・ノードを含む) 上の、コンテナ化された、あるいはローカルにインストールされた、任意の InterSystems IRIS インスタンスで使用できます。

コンテナの導入に使用される構成マージ・ファイルの例は、"自動導入での有用なパラメータ" を参照してください。"構成パラメータ・ファイル・リファレンス" には、すべての InterSystems IRIS 構成パラメータの包括的な説明が含まれています。

2 InterSystems IRIS を構成する方法

InterSystems IRIS インスタンスの構成は、インストール・ディレクトリにある、名前と値のペアの構成パラメータを含む、iris.cpf という名前のファイルで指定されます。導入後に初めて起動する場合も含め、インスタンスは、起動するたびにこの構成パラメータ・ファイル (CPF) を読み取って設定値を取得します。このため、CPF を変更してインスタンスを再起動すれば、いつでもインスタンスを再構成できます。

例えば、CPF の `[config]` セクションにある globals 設定は、インスタンスのデータベース・キャッシュのサイズを指定します。新たにインストールされたインスタンスの CPF の設定では、初期キャッシュ・サイズは合計システム・メモリの 25% に指定されています。これは、実稼働環境での使用を対象としていません。データベース・キャッシュのサイズを変更するには、インスタンスの CPF を任意のテキスト・エディタで開き、目的のキャッシュ・サイズを globals の値として指定してインスタンスを再起動します。ほとんどのパラメータは他の方法でも変更できます。例えば、globals の値は、管理ポータルを使用するか、永続クラス Config.config のメソッドを使用して変更できます。ただし、1 つのインスタンスに 1 回の操作で複数の構成変更を加える場合、および複数のインスタンスの同時構成を自動化する場合、これを実行できる一般的な方法はインスタンスの CPF を更新する方法だけです。

CPF の使用と内容については、"構成パラメータ・ファイル・リファレンス" を参照してください。

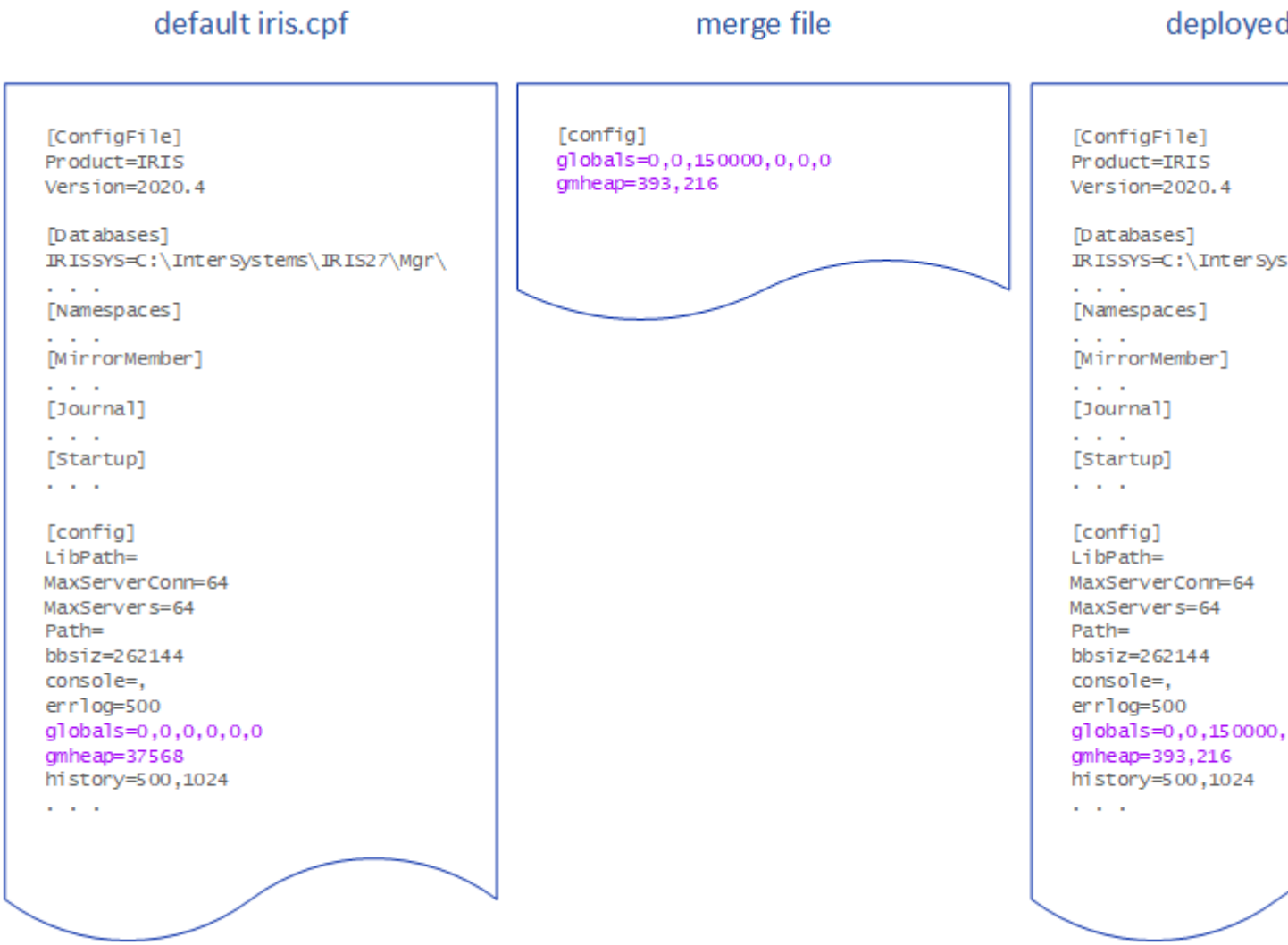
3 構成マージの仕組み

構成マージ・ファイルは、必要な InterSystems IRIS 構成パラメータおよび値のサブセットを含む部分的な CPF です。構成マージでマージ・ファイルがインスタンスに適用されると、CPF を手動で編集して値を変更しているかのように、これら

の設定がインスタンスの CPF にマージされ、値が置き換えられます。元の CPF に存在しないパラメータがマージ・ファイルにある場合、そのパラメータは適切な場所に追加されます。

例えば、シャード・クラスタ内のデータ・ノードと計算ノードは通常、他の目的のために構成されたデータベース・キャッシュよりもはるかに大きいデータベース・キャッシュを必要とし、共有メモリもより多く構成されます。導入時に大きなデータベース・キャッシュと多くの共有メモリを持つようにインスタンスを構成するには、あるいは既存のインスタンスをこのように再構成するには、globals パラメータ (データベース・キャッシュのサイズを指定) と gmheap パラメータ (共有メモリの容量を指定) に目的の値を指定した構成マージ・ファイルを適用します。インスタンスの CPF 内の既定値がこれらの値で置換されます。以下に、インスタンスの導入時に、マージ・ファイルを使用してこれらのパラメータを更新する方法について説明します。

図 1: 導入時のマージ・ファイルによるメモリ設定の更新



4 構成マージで、構成以上のカスタマイズは可能か

構成パラメータの値の変更に加え、構成マージによって、多種多様な InterSystems IRIS オブジェクト (ネームスペースおよびデータベース、ユーザ、ロール、リソースなど) や、ミラーおよびミラー・メンバを作成、変更、削除できます。これは、[Actions] セクションにあるパラメータを使用して実行します。このセクションは、マージ・ファイル内でのみ有効で、インスタンスの CPF には表示されません (追加することもできません)。例えば、インスタンスにグローバル・マッピングを

追加するには、マージ・ファイルに CreateMapGlobal パラメータを含む [Actions] セクションを含めます。システム・オブジェクトを作成、変更、削除するこのセクションのパラメータは、更新パラメータと呼ばれるパラメータ値を更新するパラメータと区別するために、アクション・パラメータと呼ばれることがあります。

アクション・パラメータを使用して、新しいインスタンスと既存のインスタンスの両方でオブジェクトを管理できます。アクション・パラメータで指定された操作はベキ等であり、変更が加えられる場合にのみ実行されます。具体的には、以下のとおりです。

- 指定したオブジェクトが存在する場合、作成アクションは実行されません。
- 指定したオブジェクトが存在しない場合、変更アクションは実行されません。(オブジェクトは存在するが、アクションがそのオブジェクトのプロパティに追加されない/そのプロパティを変更しない場合、アクションは実行されますが効果はありません。)
- 指定したオブジェクトが存在しない場合、削除アクションは実行されません。
- オブジェクトが存在し、かつアクションがそのオブジェクトのプロパティに追加されない/そのプロパティを変更しない場合、構成アクションは実行されません。オブジェクトが存在しない場合、またはオブジェクトが存在し、アクションがプロパティに追加される/そのプロパティを変更する場合、アクションは実行されます。

アクションのパラメータは導入時に非常に有用です。例えば、ConfigMirror アクションと、CreateDatabase アクションの MirrorSetName および MirrorDBName プロパティを使用して、導入された複数のインスタンスをミラーとして構成し、導入後すぐに、ミラーリングされたデータベースを配置して、新しいミラーを完全に機能させることができます。一方、iris merge コマンドを使用した既存のインスタンスの再構成では、アクション・パラメータにより、直ちに変更を加えることができます。このパラメータを使用しない場合は、管理ポータルでの手順またはクラス・メソッドの呼び出しが必要になります。主な例としては、ModifyDatabase アクションの MirrorSetName および MirrorDBName プロパティを使用し(実行中のプライマリ・インスタンスで行う必要があります)、既存のミラーに新しいデータベースを追加することで、既存のミラーにアービターを追加するなどが挙げられます。

アクション・パラメータによって実行される操作は、Config および Security クラスと、SYS.Database および %SYSTEM.SQL.Security クラスのクラスのメソッド、および 2 つの SQL コマンドを呼び出すことで制御されます。

アクション・パラメータの使用例は、"自動導入での有用なパラメータ" を参照してください。

管理されるオブジェクト別、対応するクラス別、処理順、および名前別のアクション・パラメータのリストは、[Actions] パラメータ・リファレンスを参照してください。

5 構成マージの使用法

構成マージの主な用途としては、以下の 2 つがあります。

- 導入時の複数インスタンス・トポロジとスタンドアロン・インスタンスの構成
- 導入された複数インスタンス・トポロジとスタンドアロン・インスタンスの再構成

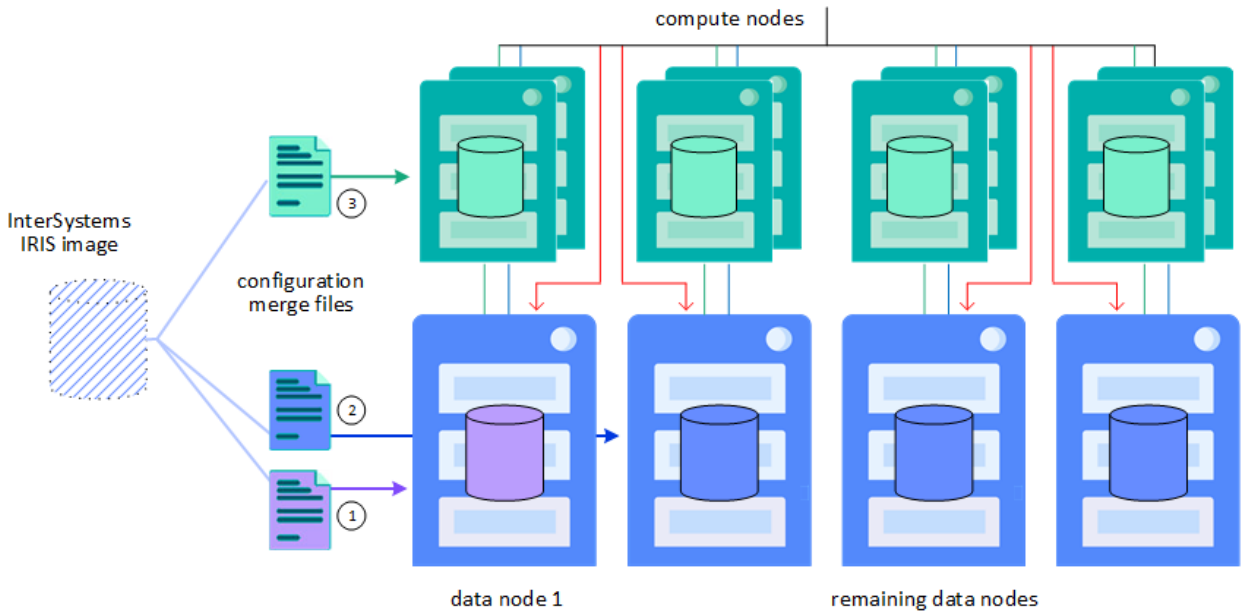
構成マージ機能のアプリケーションに関係なく、関係するマージ・ファイルをバージョン管理下に置き、インスタンスまたは複数インスタンス・トポロジの導入以降の継続期間を通して、すべての構成変更のレコードを提供することをお勧めします。

自動導入または再構成プロセスに構成マージを組み込むと、適用するマージ・ファイルを更新するだけでそのプロセスを更新できます。開発やテストなどの目的で使用される個々のインスタンスにおいても、ユーザはインスタンスの導入または再構成の前に適切なマージ・ファイルの最新バージョンを取得し、その構成を一元的な仕様に確実に一致させる必要がある場合があります。バージョン管理により、以前のバージョンのマージ・ファイルを選択して、古い構成に戻すこともできます。

6 導入での構成マージの使用法

導入時に構成マージ・ファイルを適用することにより、導入したインスタンスを初めて開始する前に、その既定の構成を変更できます。これにより、同じイメージからさまざまな構成でコンテナを導入したり、同じキットから構成の異なるインスタンスを直接複数インスタンス・トポロジにインストールすることができ、導入後に目的のトポロジにインスタンスを構成する必要はありません。例えば、計算ノードを含むシャード・クラスタのコンテナ化された自動導入では、次の図に示すように、データ・ノード 1、残りのデータ・ノード、および計算ノードに、この順序で異なるマージ・ファイルを適用することができます。すべてのインスタンスが起動および実行されると、シャード・クラスタも起動され、実行されます。

図 2: 構成マージを使用したシャード・クラスタの自動導入



同様に、ミラーを導入する場合、プライマリ、バックアップ、および非同期メンバに対して異なる構成マージ・ファイルを適用できます。ミラーリングされたシャード・クラスタでも、このアプローチを使用して簡単に導入できます。

構成マージを有効にするには、導入時に、環境変数 **ISC_CPF_MERGE_FILE**、または InterSystems IRIS の自動化された導入ツール、InterSystems Cloud Manager (ICM) または InterSystems Kubernetes Operator (IKO) のいずれかのこの目的に使用されるフィールドによってマージ・ファイルの場所を指定するだけです。例えば、手動またはスクリプト化された導入で、以下のように指定します。

```
ISC_CPF_MERGE_FILE=/home/user/mergefiles/cmf_090821.cpf
```

マージ・ファイルを指定する具体的な方法は、使用する導入メカニズムと、インスタンスがコンテナ化されているかどうかによって異なります。

- InterSystems IRIS コンテナの導入
- キットからの InterSystems IRIS のインストール
- InterSystems Cloud Manager を使用した導入
- InterSystems Kubernetes Operator を使用した導入

6.1 マージ・ファイルを使用した InterSystems IRIS コンテナの導入

InterSystems IRIS コンテナを導入する際、以下の方法で環境変数とマージ・ファイルを含めることができます。

- 導入に使用しているスクリプトまたは docker-compose.yml ファイルに含めます。

  以下のサンプル導入スクリプトでは、**ISC_CPF_MERGE_FILE** で指定されているマージ・ファイルとライセンス・キーは、**ISC_DATA_DIRECTORY** で永続的な %SYS に指定された外部ボリュームにステージングされるため、コンテナ内部でアクセス可能になります。

```
#!/bin/bash
# script for quick demo and quick InterSystems IRIS image testing

# Definitions to toggle_____________________________________________
container_image="intersystems/iris-arm64:2021.1.0.205.0"

# the docker run command

docker run -d
  -p 9091:1972
  -p 9092:52773
  -v /data/durable263:/durable
  -h iris
  --name iris
  --cap-add IPC_LOCK
  --env ISC_DATA_DIRECTORY=/durable/irisdata
  --env ISC_CPF_MERGE_FILE=/durable/merge/CMF.cpf
  $container_image
  --key /durable/key/iris.key
```

  このサンプルの docker-compose.yaml ファイルには、導入スクリプトと同じ要素が含まれます。

```
version: '3.2'

services:
  iris:
    image: intersystems/iris-arm64:2021.1.0.205.0
    command: --key /durable/key/iris.key
    hostname: iris

    ports:
    # 1972 is the superserver default port
    - "9091:1972"
    # 52773 is the webserver/management portal port
    - "9092:52773"

    volumes:
    - /data/durable263:/durable

    environment:
    - ISC_DATA_DIRECTORY=/durable/irisdata
    - ISC_CPF_MERGE_FILE=/durable/merge/CMF.cpf
```

- マージ・ファイルをコンテナ・イメージに含め、環境変数を導入に使用されるスクリプトまたは Docker Compose ファイルに含めます。

  インターシステムズ提供の InterSystems IRIS イメージによって開始し独自のコードと依存関係を追加することでカスタムの InterSystems IRIS コンテナ・イメージを作成する場合、Dockerfile で iris merge コマンドを実行し、イメージに含まれている InterSystems IRIS インスタンスを再構成できます。例えば、ネームスペースとデータベースをインスタンス（カスタム・イメージから作成した各コンテナ内のインスタンスに存在する）に追加する `[Actions]` パラメータを含むマージ・ファイルで、iris merge コマンドを実行します。このメソッドは、"コンテナ内でのインターシステムズ製品の実行" の "InterSystems IRIS イメージの作成" で説明されています。

  コンテナ化されたインスタンスがコンテナに配置したマージ・ファイルを使用し続けるようにするには、スクリプトに **ISC_CPF_MERGE_FILE** を設定するか、このファイルの場所にファイルを作成します。永続的な SYS ボリュームに配置されたマージ・ファイルを使用して、上記のように、導入時に追加で別個のマージを実行することもできます。これを行う場合は、含まれているマージ・ファイルをインスタンスに適用した後、このファイルを削除するよう、Dockerfile にコマンドを追加します。

構成マージを使用した自動導入のユース・ケースの例は、"自動導入での有用なパラメータ" を参照してください。

6.2 マージ・ファイルを使用したキットからの InterSystems IRIS のインストール

キットから InterSystems IRIS をインストールする際、マージ・ファイルを手動またはスクリプトで適用するには、以下の手順を使用してインストールと起動を分離する必要があります。

1. 以下のように、irisinstall または irisinstall_silent スクリプトの前に **ISC_PACKAGE_STARTIRIS** パラメータを付けて、インスタンスを起動せずにインストールします。

```
ISC_PACKAGE_INSTANCENAME="IRIS27" ISC_PACKAGE_STARTIRIS="N" /tmp/iriskit/irisinstall
```

2. 以下のように、前に **ISC_CPF_MERGE_FILE** 変数を付けた iris start コマンドでインスタンスを起動します。

```
ISC_CPF_MERGE_FILE=/tmp/iriskit/CMF/merge.cpf iris start IRIS27
```

6.3 InterSystems Cloud Manager を使用した導入時のマージ・ファイルの使用法

InterSystems Cloud Manager (ICM) は、InterSystems IRIS のエンド・ツー・エンドのプロビジョニングおよび導入ソリューションです。ICM を使用すると、Google Cloud Platform、Amazon Web Services、Microsoft Azure などのパブリック・クラウド・プラットフォーム、またはプライベート VMware vSphere クラウドで、あるいは、既存の仮想システムまたはハードウェア・システムで、インフラストラクチャをプロビジョニングし、コンテナ化された InterSystems IRIS ベースのサービスを導入できます。ICM では、インターシステムズ製のコンテナと共にカスタムのあるいはサードパーティ製のコンテナを導入できます。また、InterSystems IRIS キットからコンテナレス・インストールを実行することもできます。

ICM では、UserCPF プロパティを通じて構成マージ機能を利用可能です。このプロパティを構成ファイル (defaults.json または definitions.json) に追加して、適用するマージ・ファイルの場所を指定できます。既定ファイルに UserCPF を追加すると、同じマージ・ファイルが導入されたすべてのインスタンスに適用されます。definitions.json 内のノード定義にこれを追加すると、異なるノード・タイプに異なるマージ・ファイルを適用できます。

ICM での構成マージの使用の詳細は、"InterSystems Cloud Manager ガイド" の "カスタマイズされた InterSystems IRIS 構成を使用した導入" を参照してください。

6.4 InterSystems Kubernetes Operator による導入時のマージ・ファイルの使用

Kubernetes は、コンテナ化されたワークロードとサービスの導入、拡張、および管理を自動化するためのオープンソースのオーケストレーション・エンジンです。InterSystems Kubernetes Operator (IKO) は、IrisCluster カスタム・リソースで Kubernetes API を拡張します。このリソースは、InterSystems IRIS のシャード・クラスタ、分散キャッシュ・クラスタ、またはスタンドアロン・インスタンスとして、すべて任意でミラーリングした状態で、Kubernetes プラットフォームに導入できます。IKO は、InterSystems IRIS 固有のクラスタ管理機能も Kubernetes に追加して、クラスタにノードを追加するなどのタスクの自動化を可能にします。このようなタスクは、自動化されなければ、インスタンスを直接操作して手動で行わなければなりません。

IKO を使用して導入する場合は、Kubernetes ConfigMap を使用して 1 つ以上のマージ・ファイルを導入プロセスに統合します。詳細は、"InterSystems Kubernetes Operator の使用" の "構成ファイルの作成とその ConfigMap の提供" を参照してください。

7 構成マージを使用して既存のインスタンスを再構成する方法

実行している複数のインスタンスに対する同じマージ・ファイルの適用を自動化することによって、これらのインスタンスすべてを同時に同じ方法で再構成でき、アプリケーションまたはクラスタ全体にわたって同じ一連の構成変更を適用できます。インスタンスごとにカスタマイズされている可能性があるため変更すべきではない設定がある場合は、それらの設定を更新しないようにすることができます。その場合、該当する設定をマージ・ファイルから除外し、変更しても安全で、

変更することが望ましいとわかっている設定のみを含めます。前のセクションで説明したとおり、自動化された 1 つのプログラムを使用して、インスタンスのグループごとに（異なるミラー・メンバやクラスタ・ノード・タイプなど）異なるマージ・ファイルを提供することもできます。

マージ・ファイルを使用したすべての構成変更の適用は、変更のプロセスを合理化するのに役立ち、インスタンスの構成に対する高度な制御を維持できます。ターミナルからの多くの個別の変更を、管理ポータルの複数のページで、またはインスタンスの CPF を手動で編集することによって行うのではなく、マージ・ファイル内の同一の構文を使用して、一度にすべての変更を実行することができます。マージ・ファイルをバージョン管理下に置くことで、構成履歴や以前の構成のリストアのオプションを確実に利用できます。

iris merge コマンドは、マージ・ファイルを実行しているインスタンスに適用します。以下のように実行されます。

```
iris merge instance [merge-file] [target-CPF]
```

説明：

- instance は、InterSystems IRIS インスタンスの名前です。
- merge-file は、ファイル名を含む、マージ・ファイルのパスです。merge-file が指定されていない場合は、**ISC_CPF_MERGE_FILE** 環境変数の値が使用されます（設定されている場合）。
- target-CPF は、instance インスタンスのアクティブな CPF の場所で、iris.cpf という名前であると見なされます。target-CPF が指定されていない場合、既定では以下のとおりです。
  - コンテナ化されていないインスタンスの場合、**ISC_PACKAGE_INSTALLDIR** 環境変数（設定されている場合）で指定されたディレクトリにある iris.cpf ファイル。ほとんどの既存のインスタンスでは、この変数は設定されていないため、対象の CPF の場所を明示的に指定する必要があります。
  - コンテナ化されたインスタンスの場合、**ISC_DATA_DIRECTORY** 環境変数で指定されたディレクトリにある iris.cpf ファイル。この変数が設定されていない（永続的な %SYS が使用されていないため）場合は、**ISC_PACKAGE_INSTALLDIR** 環境変数（この変数は常に InterSystems IRIS コンテナに設定されています）。

以下の場合、マージは実行されません。

- 指定したマージ・ファイルが存在しない場合、または merge-file 引数が省略され、**ISC_CPF_MERGE_FILE** が存在しない場合。
- 指定した対象の場所に CPF がない場合、または対象の場所が指定されておらず、**ISC_DATA_DIRECTORY** も **ISC_PACKAGE_INSTALLDIR** も存在しない場合。

終了コード 3 は、マージが成功したことを示します。

CPF にマージされる一部の変更はすぐには適用されず、再起動が必要です。例えば、インスタンスの共有メモリ・ヒープのサイズを指定する gmheap パラメータの値の変更は、そのインスタンスが再起動されるまで適用されません。マージ・ファイルにそのようなパラメータが 1 つ以上含まれる場合、次のサンプル・スクリプトの抜粋のように、再起動の一部としてマージ・ファイルを適用する必要が生じることがあります。

```
# restart instance with  the necessary parameters (all on one line)
sudo ISC_CPF_MERGE_FILE=/net/merge_files/config_merge.cpf iris stop IRIS restart
```

一方、iris merge コマンドを使用してマージ・ファイルを適用すると、再起動しなくても設定（インスタンスの起動時に設定できないものを含む）をすぐに変更できます。"構成マージで、構成以上のカスタマイズは可能か" に記載している例では、既存のミラーにデータベースを追加しています。

重要　コンテナが構成マージで導入される場合（"マージ・ファイルを使用した InterSystems IRIS コンテナの導入" の説明に従って）、コンテナが実行されている限り、**ISC_CPF_MERGE_FILE**（コンテナ内で永続）で指定されたマージ・ファイルの更新が継続的に監視され、更新が発生すると iris merge コマンドによって直ちにマージされます。つまり、マージ・ファイルを更新することで、いつでもコンテナ化されたインスタンスの構成を更新でき、コンテナ化されたインスタンスとクラスタの再構成を容易に自動化できます。

8 構成変更の管理

導入において、または既存のインスタンスで、iris merge コマンドにより、または再起動時に構成マージを使用することに加え、管理ポータル、Config.* クラス、またはテキスト・エディタを使用してインスタンスの CPF を変更することができます。これらの方法は通常、複数のインスタンスを再構成する場合ではなく、必要に応じて個々のインスタンスの設定を個別に変更する場合に使用します。複数のインスタンスを自動的に導入したり再構成する場合、推奨されるベスト・プラクティスは、すべての構成の変更をこの方法に限定することです。これは、例えば iris merge マージを使用して 1 つのインスタンスの 1 ～ 2 個のパラメータのみを変更するような場合にも当てはまります。そのようにして、使用するマージ・ファイルのバージョンを管理して保存することで、各インスタンスの構成のレコードを時系列的に維持できると共に、構成マージ以外の方法で加えた変更を構成マージで上書きしてしまう可能性を回避できます。

コンテナでは、**ISC_CPF_MERGE_FILE** 変数によって指定されたマージ・ファイルの継続的な監視およびマージにより、後者の可能性が非常に大きくなります（前のセクションの説明を参照）。これにより、構成マージとマージ・ファイルの一元管理リポジトリを使用すると、マージ・ファイルを更新するだけで、いつでも既存のインスタンスにさらに変更を適用することができます。ただし、マージ・ファイルに含まれる構成パラメータが導入時以降にコンテナ内のインスタンス上で別の方法で変更されている場合、更新マージによってこれらの変更が消去されてしまい、変更のレコードがまったく残らない可能性があります。すべての構成変更を構成マージに限定することで、これを回避できます（マージ・ファイルが存在しない場合は、起動時にエラー・メッセージが表示され、続行されます）。

変更を構成マージに限定しない場合、インスタンスの起動後に、以下のスクリプトのどちらかまたは両方を（例えば iris-main --after オプションを使用して）自動化に含めることで、構成マージで不要な変更が行われる可能性を回避できます。

- 導入済みの各コンテナ内の **ISC_CPF_MERGE_FILE** 環境変数を削除する。（マージ・ファイルが存在しない場合は、起動時にエラー・メッセージが表示され、続行されます。）
- 各コンテナ内のマージ・ファイルを空のファイルで置き換える。

重要　ICM で UserCPF パラメータによって指定された構成マージ・ファイルを使用して、導入するコンテナをカスタマイズする場合、"ICM ガイド" の "カスタマイズされた InterSystems IRIS 構成を使用した導入" で説明しているように、導入後にコンテナからマージ・ファイルが自動的に削除されますが、**ISC_CPF_MERGE_FILE** 環境変数は削除されません。

9 自動導入での有用なパラメータ

構成マージ機能を使用して、インスタンスの CPF 内にある設定を任意の組み合わせで更新し、`[Actions]` セクションで指定されているとおりにインスタンス上で特定の操作を実行することができます。いくつかの自動導入のユース・ケースは有用で、構成マージ機能と関連するパラメータの高度な性能を示す良い例となっています。このセクションでは、以下のようなこれらのユース・ケースについて説明します。

更新パラメータ

- 既定のパスワードの変更

- メモリの構成と割り当て
- SQL および SQL シェル・オプションの構成と SQL データ型のマッピング
- 更新パラメータの例

アクション・パラメータ

- セキュリティ・オブジェクトの作成、変更、削除
- データベース・オブジェクトの作成、変更、削除
- 分散キャッシュ・クラスタの導入
- クラスタのデータ・サーバのミラーリング

9.1 更新パラメータ

以降のセクションで説明するパラメータは、導入されたインスタンスを起動する前に、インスタンスの CPF の値を変更して、導入ソース (インストール・キットまたはコンテナ) の既定の CPF を更新するために使用するパラメータです。以下の各パラメータ名は、"構成パラメータ・ファイル・リファレンス" 内のリストにリンクされているため、パラメータの目的とその使用法の詳細を容易に参照できます。

9.1.1 既定のパスワードの変更

"コンテナ内でのインターシステムズ製品の実行" の "認証とパスワード" で説明しているように、[Startup] セクションの PasswordHash 設定を使用して、導入時にインスタンスの事前定義アカウントの既定のパスワードをカスタマイズすることができます。そうすることで、**SYS** の既定のパスワードを有効なままにしておくことに伴う重大なセキュリティ・リスクを排除できます(各事前定義アカウントのパスワードは、導入後に個別に変更する必要があります)。

テーブル 1: Password パラメータ

[Startup] パラメータ	指定する内容
PasswordHash	値の暗号化ハッシュとそのソルトに基づく、事前定義のユーザ・アカウントの既定のパスワード

注釈　[Actions]/CreateUser パラメータは、パラメータと同等の PasswordHash 引数も取ります ("[Actions] パラメータ・リファレンス" を参照)。

9.1.2 メモリの構成と割り当て

InterSystems IRIS インスタンスのメモリ使用率に影響するパラメータは多数あり、その最適な値は使用可能な物理メモリ、クラスタ内でのインスタンスのロール、関連するワークロード、およびパフォーマンスの要件によって異なります。

例えば、インスタンスのデータベース・キャッシュの最適なサイズ (globals パラメータを使用して指定可能) は、インスタンスのロールによって大きく異なる可能性があります。前述のように、通常、シャード・クラスタのデータ・ノードには比較的大きなキャッシュが必要です。しかしそのロール内であっても、最適なサイズはクラスタのシャード・データ・セットのサイズによって異なり、さらに実装されるサイズはリソースの制約のために最適な値よりも小さくなる可能性があります (詳細は、"スケーラビリティ・ガイド" の "InterSystems IRIS シャード・クラスタの計画" を参照してください)。また、データベース・キャッシュのサイズ設定は通常慎重に行う必要があるため、既定のデータベース・キャッシュ設定 (コンテナで提供される iris.cpf ファイル内の globals の値) は、インスタンスのロールに関係なく、意図的に実稼働環境に適さないものになっています。

以下の表に、CPF の [Config] セクション内にある、導入の一環として更新できるメモリ使用率のいくつかの設定を示します。

テーブル 2: メモリ・パラメータ

[Config] パラメータ	指定する内容
bbsiz	プロセスあたりのプロセス・プライベート・メモリの最大値
globals	データベース・キャッシュに割り当てられる共有メモリ (共有メモリ・ヒープからではない)
routines	ルーチン・キャッシュに割り当てられる共有メモリ (共有メモリ・ヒープからではない)
gmheap	共有メモリ・ヒープとして構成される共有メモリ
jrnbufs	共有メモリ・ヒープからジャーナル・バッファに割り当てられる共有メモリ
locksiz	共有メモリ・ヒープからロックに割り当てられる共有メモリの最大値

これらのパラメータおよびメモリ関連のその他のパラメータの詳細は、"システム・リソースの計画と管理"、"メモリと開始設定"、"ジャーナル設定の構成"、および "ロックの監視" を参照してください。

9.1.3 SQL および SQL シェル・オプションの構成と SQL データ型のマッピング

CPF の `[SQL]` セクションにあるパラメータを 1 つ以上マージすることによって、導入するインスタンスの SQL および SQL シェルの設定を指定できます。管理ポータルの [SQL] ページ ([**システム管理**]→[**構成**]→[**SQL とオブジェクトの設定**]→[**SQL**]) で、これらの設定を確認および変更できます。CPF の `[SqlSysDatatypes]` および `[SqlUserDatatypes]` セクションを使用して、SQL システム・データ型と SQL ユーザ・データ型を、導入されたインスタンス上で InterSystems SQL の同等のデータ型にマッピングできます。SQL シェルの設定およびデータ型マッピングの詳細は、それぞれ "SQL シェルの構成" と "データ型 (SQL)" を参照してください。

9.1.4 更新パラメータの例

以下のサンプル CPF マージ・ファイルには、前のセクションで説明したいくつかの更新パラメータが含まれています。`SystemMode` パラメータは、管理ポータルの上部に表示されるラベルを指定します。

```
[Startup]
SystemMode=TEST
PasswordHash=FBFE8593AEFA510C27FD184738D6E865A441DE98,u4ocm4qh

[config]
bbsiz=-1
globals=0,0,900,0,0,0
routines=64
gmheap=256000
jrnbufs=96
locksiz=1179648

[SQL]
DefaultSchema=user
TimePrecision=6

[SqlSysDatatypes]
TIMESTAMP=%Library.PosixTime
```

9.2 アクション・パラメータ

以降のセクションで説明するパラメータは、導入 (または再構成) の一環として、インスタンスのさまざまなタイプのオブジェクト (データベース、ネームスペース、マッピングなど) や、ユーザ、ロール、リソースなどを作成、変更、削除するために `[Actions]` セクションに含めることのできるパラメータです。アクション・パラメータ (多くの場合、単にアクションと呼ばれます) の使用法は、"構成マージで、構成以上のカスタマイズは可能か" に説明されています。このセクションには、その他の構成マージ・ファイルの例が記載されており、"[Actions] パラメータ・リファレンス" にこれらの完全なリストが示されています。

すべての [Actions] パラメータのリストは、"[Actions] パラメータ・リファレンス" を参照してください。

9.3 セキュリティ・オブジェクトの作成、変更、削除

導入または再構成の一環として、セキュリティ・オブジェクトを作成および変更するには、以下の表で説明する操作を [Actions] セクションに含めます。

テーブル 3: セキュリティ・オブジェクト作成パラメータのサンプル

[Actions] パラメータ	指定する内容
CreateUser	作成するユーザ・アカウントの名前とプロパティ。ModifyUser および DeleteUser も使用できます。
CreateRole	作成するロールの名前とプロパティ。ModifyUser および DeleteUser も使用できます。
CreateResource	作成するリソースの名前とプロパティ。ModifyUser および DeleteUser も使用できます。
GrantAdminPrivilege GrantPrivilege	SQL 特権と SQL 管理特権を付与するユーザ・アカウント、付与する特権、およびそれらを付与するネームスペース。RevokeAdminPrivlege および RevokePrivilege も使用できます。
ModifyService	有効または無効にするサービス。
CreateApplication	作成するアプリケーションの名前とプロパティ。ModifyApplication および DeleteApplication も使用できます。
CreateSSLConfig	作成する TLS/SSL 構成の名前、場所、およびプロパティ。ModifySSLConfig および DeleteSSLConfig も使用できます。
CreateLDAPConfig	作成する LDAP 構成の名前とプロパティ。ModifyLDAPConfig および DeleteLDAPConfig も使用できます。
CreateEvent	作成するシステム監査イベントの名前、プロパティ、およびステータス。ModifyEvent および DeleteEvent も使用できます。

セキュリティ・オブジェクトでのアクション・パラメータの使用法を説明するために、導入されたインスタンスに、以下の SQL 管理者ユーザの事前定義アカウントを追加するとします。

- %Service_Bindings サービス (%SQL ロール) を介した SQL アクセスを持っている。
- **USER** データベースに対する読み取りまたは書き込みが可能 (%DB_USER ロール)。
- テーブル、ビュー、プロシージャ、関数、メソッド、クエリ、トリガを作成および削除でき (%DB_OBJECT_DEFINITION 特権)、**USER** ネームスペースで BUILD INDEX コマンドを使用できる (%BUILD_INDEX 特権)。

これを行うために、以下のように、CreateUser パラメータを使用してパスワードを持つユーザ・アカウントを作成して必要なロールを割り当て、GrantAdminPrivilege パラメータを使用してこれに必要な SQL 特権を付与できます。

```
[Actions]

CreateUser:Name=SQLAdmin,
  PasswordHash="cec6638a357e7586fddfb15c0e7dd5719a1964e774cd37466fb0c49c05,
  323cb89148c887166dd2be61c107710539af2c01b43f07dccc8d030ac2c1a8cf7c5ace4a00d57e3780f,10000,SHA512",
  Roles="%SQL,%DB_USER"

GrantAdminPrivilege:Grantee=SQLAdmin,Namespace=USER,AdminPriv="%DB_OBJECT_DEFINITION,%BUILD_INDEX"
```

これらのアクション・パラメータによって実行される操作と、そのプロパティの値の詳細は、"認証とパスワード"、"インターシステムズの承認について"、および "SQL のユーザ、ロール、および特権" を参照してください。

9.4 データベース・オブジェクトの作成、変更、削除

導入または再構成の一環として、データベース (ローカルとリモートの両方)、ネームスペース、マッピングを作成するには、以下の表で説明する操作を `[Actions]` セクションに含めます。

テーブル 4: データベースおよびネームスペースのアクションのパラメータのサンプル

[Actions] パラメータ	指定する内容
`CreateDatabase`	InterSystems IRIS に登録するデータベースの名前とプロパティ、および作成するデータベース・ファイルのホスト・ファイル・システム上の場所。`ModifyDatabase` および `DeleteDatabase` も使用できます。
`CreateDatabaseFile`	(データベースを InterSystems IRIS に登録せずに) 作成するデータベース・ファイルのホスト・ファイル・システム上の場所。`ModifyDatabaseFile` および `DeleteDatabaseFile` も使用できます。
`CreateNamespace`	InterSystems IRIS で作成するネームスペースの名前とプロパティ。`ModifyNamespace` および `DeleteNamespace` も使用できます。
`ModifyNamespace`	変更する既存の InterSystems IRIS ネームスペースの名前とそのプロパティ。`CreateNamespace` および `DeleteNamespace` も使用できます。
`CreateMapGlobal`	マッピングを作成するネームスペース、マッピングするグローバルの仕様、そのグローバルが存在するデータベース。`ModifyMapGlobal` および `DeleteMapGlobal` も使用できます。さらに、`Create/Modify/DeleteMapRoutine` および `Create/Modify/DeleteMapPackage` を使用して、ルーチンおよびパッケージのマッピングを作成、変更、削除できます。

データベース関連オブジェクトでのアクション・パラメータの使用法を説明するために、以下を行うとします。

- データベースとそのリソースを作成してから、既存のネームスペースを変更し、作成したデータベースをグローバル・データベースにして相互運用性を有効にします。
- 2 番目のデータベースとリソースを作成し、そのデータベースを既定のグローバル・データベースにして、相互運用対応ネームスペースを作成します。

以下の例は、マージ・ファイルでこれを行う方法を示しています。

```
[Actions]

CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Directory=/database-path/APPA
ModifyNamespace:Name=APPA,Globals=APPA,Interop=1

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Directory=/database-path/APPB
CreateNamespace:Name=APPB,Globals=APPB,Interop=1
```

後から、データベース **APPA** 内のグローバルおよびルーチンのネームスペース **APPB** へのマッピングを追加するとします。これは、マージ・ファイルで以下のように行うことができます。

```
[Actions]
CreateMapGlobal:Namespace=APPB,Name=global-spec,Database=APPA
CreateMapRoutine:Namespace=APPB,Name=routine-spec,Database=APPA
```

これらのアクション・パラメータによって実行される操作と、そのプロパティの値の詳細は、"ネームスペースの作成/変更"、"ローカル・データベースの作成"、および "ネームスペースへのグローバル、ルーチン、およびパッケージ・マッピングの追加" を参照してください。

9.5 分散キャッシュ・クラスタの導入 (ミラーリングなし)

前のセクションのマージ・ファイルの例に簡単な変更をいくつか加えることで、分散キャッシュ・クラスタを導入するためのマージ・ファイルを作成できます。

データ・サーバの導入

以下のマージ・ファイルを使用して、ミラーリングされていないデータ・サーバを導入します。このマージ・ファイルは、アプリケーション・データベースと関連するリソースおよびネームスペースの作成に加え、以下を行います。

- アクション・パラメータを使用して ECP サービスを有効にします。
- 更新パラメータを使用して、データ・サーバが受け入れることができる同時アプリケーション・サーバ接続の最大数を 16 に設定します。

前のサンプル・マージ・ファイルとの違いが強調表示されています。

```
# nonmirrored data server merge file

[Config]

MaxServerConn=16

[Actions]

ModifyService:Name=%service_ecp,Enabled=1

CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Directory=/database-path/APPA
CreateNamespace:Name=APPA,Globals=APPA

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Directory=/database-path/APPB
CreateNamespace:Name=APPB,Globals=APPB
```

アプリケーション・サーバの導入

すべてのアプリケーション・サーバの導入に使用されるこのマージ・ファイルは、以下を行います。

- 更新パラメータを使用して、データ・サーバをリモート・サーバとして追加します。
- `Server` および `LogicalOnly` プロパティを追加し、ローカル・データベースを作成するローカル・ディレクトリではなくリモート・サーバ上の既存のデータベースを指すように `Directory` プロパティを更新して、上述の `CreateDatabase` アクションを変更します。

前のセクションのサンプル・マージ・ファイルとの違いが強調表示されています。

```
# app servers merge file

[ECPServers]

dataAB=dataserver-address,port,0

[Actions]

CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Server=dataAB,Directory=/database-path-on-dataserver-dataAB/APPA,ResourceName=%DB_%APPA,
   LogicalOnly=1,
CreateNamespace:Name=APPDBA,Globals=APPA

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Server=dataAB,Directory=/database-path-on-dataserver-dataAB/APPB,ResourceName=%DB_%APPB,
  LogicalOnly=1
CreateNamespace:Name=APPB,Globals=APPB
```

これらのアクション・パラメータによって実行される操作と、そのプロパティの値の詳細は、"リモート・データベース" と "分散キャッシュ・クラスタの導入" を参照してください。

9.6 クラスタのデータ・サーバのミラーリング

1 つ以上の InterSystems IRIS ミラーを導入するには、異なるミラー・ロールに対して、それぞれに `ConfigMirror` アクション・パラメータを含む別個の構成マージ・ファイルを使用して、最初のフェイルオーバー・メンバ、2 番目のフェイルオーバー・メンバ、DR 非同期メンバの順に導入します (レポート非同期メンバは、導入後にミラーに追加できます)。

1 つのマージ・ファイルとホスト名のマッチングを使用して導入することもできます。これにより、名前が必要なパターンに一致する各ホスト・セットに、どのメンバを導入するかが決まります。

このセクションでは、各アプローチの例と、`ConfigMirror` パラメータのよく使用されるプロパティを示す表を提供します。

ミラー構成の詳細は、"ミラーリングのアーキテクチャおよび計画" と "ミラーの作成" を参照してください。コンテナ化されたミラーの導入を計画する前、またはコンテナ化された既存のインスタンスをミラーに再構成する前は、必ず "InterSystems IRIS コンテナを使用したミラーリング" を参照してください。特に、フェイルオーバー用のコンテナまたは DR 非同期ミラー・メンバを起動する場合は常に ISCAgent が起動することを確認する必要があります。

9.6.1 別個のマージ・ファイルを使用したミラーの導入

別個のマージ・ファイルを使用した導入を計画する際、他のメンバを追加する前に、ミラー・プライマリとして構成されるインスタンスが実行されている必要がある点に留意してください。このため、他のインスタンスを残りのメンバとして導入する前に、このインスタンスが導入され、正常に起動されていることを確認する必要があります。

データ・サーバのミラー・メンバの導入

以下のマージ・ファイルは、以下を行うことで、分散キャッシュ・クラスタのデータ・サーバを DR 非同期メンバを持つミラーとして導入します。

- ミラーを作成し、メンバを追加する `ConfigMirror` アクション・パラメータを含めます。
- プライマリで、作成したデータベースをミラーに追加します (他のメンバに自動的に追加されます)。

前のセクションの対応するマージ・ファイルとの違いが強調表示されています。

```
# mirrored data server primary merge file

[Config]
MaxServerConn=16

[Actions]

ModifyService:Name=%service_ecp,Enabled=1

ConfigMirror:Name=CLUSTERAB,SSLDir=ssl-directory-path,
  Member=primary,Primary=localhost,ArbiterURL=address:port

CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Directory=/ddatabase-path/APPA,
   MirrorSetName=CLUSTERAB,MirrorDBName=APPA
CreateNamespace:Name=APPA,Globals=APPA

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Directory=/ddatabase-path/APPB,
   MirrorSetName=CLUSTERAB,MirrorDBName=APPB
CreateNamespace:Name=APPB,Globals=APPB

# mirrored data server backup/DR async merge file;
# for ConfigMirror Member property enter either =backup or =drasync as appropriate

[Config]
MaxServerConn=16

[Actions]

ModifyService:Name=%service_ecp,Enabled=1

ConfigMirror:Name=CLUSTERAB,SSLDir=ssl-directory-path,
  Member=backup|drasync,Primary=primary-address:,ArbiterURL=address:port

CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Directory=/ddatabase-path/APPA,
   MirrorSetName=CLUSTERAB,MirrorDBName=APPA
CreateNamespace:Name=APPA,Globals=APPA

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Directory=/ddatabase-path/APPB,
   MirrorSetName=CLUSTERAB,MirrorDBName=APPB
CreateNamespace:Name=APPB,Globals=APPB
```

ミラーリングされたデータ・サーバを持つアプリケーション・サーバの導入

このマージ・ファイルは、強調表示されているように、[ECPServers] のリモート・サーバ定義アクションの末尾の 0 を 1 に変更するだけで、上記のアプリケーション・サーバのマージ・ファイルを変更します。これは、リモート・サーバがミラーで、フェイルオーバー後にアプリケーション接続を透過的に新しいプライマリに切り替え可能であることを示します。

```
# app servers merge file

[ECPServers]

dataAB=dataserver-address,port,1

[Actions]

CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Directory=/database-path-on-dataserver-dataAB/APPA,ResourceName=%DB_%APPA,
   Server=dataAB,LogicalOnly=1
CreateNamespace:Name=APPDBA,Globals=APPA

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Directory=/database-path-on-dataserver-dataAB/APPB/,ResourceName=%DB_%APP,B
   Server=dataAB,LogicalOnly=1
CreateNamespace:Name=APPB,Globals=APPB
```

9.6.2 ホスト名マッチングを使用したミラーの導入

導入ホストの名前が -number (または正規表現 .*-[0-9]+$ として) で終了する場合 (iris-000、iris-001、iris-002 ... など)、または -number-number で終了する場合 (iris-0-0、iris-0-1、iris-1-0、iris-1-1 ... など)、1 つのマージ・ファイルから 1 つ以上のミラーを自動的に導入できます。そのためには、以下を行います。

- 必要なパターンに Map プロパティ（別個のマージ・ファイル・アプローチで使用されていない）を設定します（既定では primary,backup ですが、primary,backup,drasync,... のように最大 14 の DR 非同期メンバを含めることもできます）
- Member および Primary プロパティを auto に設定します。

例えば、以下の ConfigMirror アクション・パラメータを使用する場合、例の後の表に示すように、ミラー・メンバは適切な名前のホストに導入されます。

```
ConfigMirror:Name=AUTOMIRROR,SSLDir=ssl-directory-path,
  Map="primary,backup,drasync",
  Member=auto,Primary=auto,ArbiterURL=address:port
```

テーブル 5: ホスト名によるミラーの導入

番号が 1 つのホスト名	番号が 2 つのホスト名	ミラー・メンバのロール
mirror-000	**mirror-0-0**	プライマリ
mirror-002	**mirror-0-1**	バックアップ
mirror-003	**mirror-0-2**	DR 非同期
mirror-005	**mirror-1-0**	プライマリ
mirror-006	**mirror-1-1**	バックアップ
mirror-007	**mirror-1-2**	DR 非同期

ホスト名によるミラー・メンバの導入

前のセクションで示したように 3 つのマージ・ファイルを使用するのではなく、iris-001、iris-002、iris-003 のような連続する名前の 3 つのホストに対して以下の単一のマージ・ファイルを使用して、前述のような ConfigMirror アクションを組み込み、ホスト名マッチングを使用して 3 メンバのミラーリングされたデータ・サーバを導入できます。

```
# mirrored data server using single merge file and hostname mapping

[Config]
MaxServerConn=16

[Actions]

ModifyService:Name=%service_ecp,Enabled=1

ConfigMirror:Name=CLUSTERAB,SSLDir=ssl-directory-path,
  Map="primary,backup,drasync",Member=auto,
  Primary=auto,ArbiterURL=address:port
CreateResource:Name=%DB_%APPA,Description="APPA database"
CreateDatabase:Name=APPA,Directory=/ddatabase-path/APPA,
  MirrorSetName=CLUSTERAB,MirrorDBName=APPA
CreateNamespace:Name=APPA,Globals=APPA

CreateResource:Name=%DB_%APPB,Description="APPB database"
CreateDatabase:Name=APPB,Directory=/ddatabase-path/APPB,
  MirrorSetName=CLUSTERAB,MirrorDBName=APPB
CreateNamespace:Name=APPB,Globals=APPB
```

9.6.3 ConfigMirror のプロパティ

以下の表に、よく使用される ConfigMirror アクション・パラメータのプロパティを示します。InterSystems IRIS の以前のリリースでは、これらの多くは CPF の [Startup] セクションのパラメータでした。そのため、表には、対応する以前の [Startup] パラメータの名前を示しています。

テーブル 6: ConfigMirror のプロパティ

プロパティ	別個のマージ・ファイルを使用して導入	1 つのマージ・ファイルとホスト名マッピングを使用して導入
Name (以前は [Startup]/MirrorSetName)	新しいミラー (プライマリを導入する場合) または参加するミラー (バックアップまたは DR 非同期を導入する場合) の名前	ミラーの名前
Map	(使用されていません)	ミラー・メンバとホスト名とのマッチングに使用されるパターンを設定します。既定は Map="primary,backup" です。
Member (以前は [Startup]/MirrorMember)	導入時のミラー・メンバのロール (primary、backup、または drasync)	ミラー・メンバをホスト名に自動的にマッチングするには、auto に設定します。
Primary (以前は [Startup]/MirrorPrimary)	プライマリのホストの名前または IP アドレス	ミラー・メンバをホスト名に自動的にマッチングするには、auto に設定します。
SSLDir (以前は [Startup]/MirrorSSLDir)	インスタンスのミラー TLS/SSL 構成のホスト上の場所	(前の列を参照。TLS 構成はすべてのホスト上で同一に配置されている必要があります。)
ArbiterURL (以前は [Startup]/ArbiterURL)	ミラー (プライマリの導入時) または既存のプライマリ (バックアップまたは DR 非同期の導入時) に構成するアービターのホスト (ホスト名または IP アドレス) とポート	ミラーに構成するアービターのホスト (ホスト名または IP アドレス) とポート

10 [Actions] パラメータ・リファレンス

"構成マージで、構成以上のカスタマイズは可能か" で説明したように、[Actions] パラメータを使用すると、構成マージによって多種多様な InterSystems IRIS オブジェクトを管理できます。このセクションは、必要な [Actions] パラメータをできる限り速く識別するのに役立ちます。各表には、使用可能なすべてのパラメータが以下の順序で一覧表示されています。

- パラメータが作用するオブジェクト別。例えば、パラメータを使用してデータベースを作成、削除、変更することはできますが、サービスを作成または削除することはできません。これらは変更のみ可能です。
- パラメータが処理される順。これは、構成マージ・ファイル内の順序によって決定されるのではなく、固定されています。例えば、CreateNamespace パラメータに指定する既定のデータベースは実行時に存在している必要があるため、CreateDatabase は CreateNamespace の前に処理されます。このため、ネームスペースとその既定のグローバル・データベースまたはルーチン・データベースの両方を作成する場合、マージ・ファイルで CreateNamespace が CreateDatabase の上に表示されていても、指定したデータベースが先に作成されます。
- パラメータが対応するクラスまたはコマンド別。例えば、ミラーを作成、削除、変更するためのパラメータは、クラス Config.mirrors のメソッドの呼び出しによって制御されます。
- アルファベット順では、アクションのタイプ (作成、削除、変更) によってもパラメータを分けています。

[Actions] パラメータの引数は、次のアクションで示す形式で指定されます。ここでは、CreateNamespace パラメータを使用して、**APPDATA** データベースを既定のグローバル・データベース、**USER** データベースを既定のルーチン・データベースとして、**APPDATA** ネームスペースが作成されます。

```
[Actions]
CreateNamespace:Name=APPDATA,Globals=APPDATADB,Routines=USER
```

各 [Actions] パラメータで使用できる引数を確認するには、クラス・リファレンスでベースとなるクラスの**プロパティ・インベントリ**を参照してください。(対応するクラス別の表では、リストされるクラスは、クラス・リファレンスのエントリにリンクされています。)例えば、Config.Namespaces クラスのインベントリには、以下のプロパティが含まれます。

- Globals — ネームスペースの既定のグローバル・データベース
- Interop (ブーリアン) — ネームスペースが相互運用対応かどうか (ブーリアン)
- Routines — ネームスペースの既定のルーチン・データベース
- TempGlobals — 一時グローバル・ストレージの既定のデータベース

上述の一般的な説明には、以下のようないくつかの例外があります。

- Config パッケージのクラスから派生したすべてのパラメータの場合、オブジェクトの名前の Name プロパティは、Create()、Modify()、Delete() メソッドを含む Config.CommonMultipleMethods クラスから継承されます。
- ConfigMirrors および ConfigShardedCluster パラメータは 1 つのクラスに基づくのではなく、多数の異なる API コールを順番に呼び出します。例へのリンク
- 分散キャッシュ・クラスタ・アプリケーション・サーバで CreateDatabase パラメータを使用してリモート・データベースを追加する場合、2 つの一意の引数があり、Config.Databases プロパティ・インベントリにあるのは 1 つ目の引数のみです。
  - Server — リモート・データベースがあるデータ・サーバの名前を指定します (必須)。
  - LogicalOnly (ブーリアン) — アクションがアプリケーション・サーバ・ホストのファイル・システムでデータベースを作成しないことを指定します (オプション、既定 (0) ではデータベース・ファイルを作成します)。
- CreateUser パラメータを使用する場合、Security.Users プロパティ・インベントリにない PasswordHash 引数がハッシュ化されたパスワード、作業係数、アルゴリズムを指定します ("構成マージで、構成以上のカスタマイズは可能か" を参照)。PasswordHash が、Security.Users に一覧表示されている Password、PasswordHashAlgorithm、および PasswordHashWorkFactor プロパティの代わりに使用されます。
- SQL 特権のすべてのパラメータ — GrantPrivilege、RevokePrivilege、GrantAdminPrivilege、RevokeAdminPrivilege — は引数 Namespace を取ります。この引数はクラス・メソッドのプロパティ・インベントリ、またはコマンドを制御するためのコマンド引数リストにはありません。これらのクラスやコマンドは、呼び出しまたは実行されるネームスペースで適用されるように設計されているためです。Namespace 引数を使用すると、特権を付与または取り消す際に、任意のネームスペースを指定できます。
- SQL 管理特権パラメータ、GrantAdminPrivilege および RevokeAdminPrivilege は、クラスではなく、SQL コマンド GRANT および REVOKE に基づいており、管理者レベルの特権にのみ影響します。このため、利用可能な引数 (Namespace のほかに) には AdminPriv、Grantee、WithGrant のみが含まれます。この最後の引数は GrantAdminPrivilege 専用であり、WITH ADMIN OPTION 節を追加します (標準の WITH GRANT と同等)。
- GrantPrivilege パラメータは 2 つのメソッド %SYSTEM.SQL.Security.GrantPrivilegeWithGrant() と %SYSTEM.SQL.Security.GrantPrivilege() を組み合わせているため、追加のブーリアン引数 withGrant (wGrant も可能) を取り、withgrant=1 は前者の使用を選択し、withgrant=1 は後者の使用を選択します。

テーブル 7: オブジェクト別

Application : Create、Delete、Modify

Comport : Create、Delete、Modify

Config : Modify

CPF : Activate

Database (論理) : Create、Delete、Modify

DatabaseFile : Create、Delete、Modify

Device : Create、Delete、Modify

DeviceSubType : Create、Delete、Modify

DocDB : Create、Delete、Modify

ECP : Modify

ECPServer : Create、Delete、Modify

Event : Create、Delete、Modify

Journal : Modify

LDAPConfig : Create、Delete、Modify

LicenseServer : Create、Delete、Modify

MagTapes : Create、Delete、Modify

MapGlobal : Create、Delete、Modify

MapPackage : Create、Delete、Modify

MapRoutine : Create、Delete、Modify

MirrorMember : Modify

Mirrors : Config、Create、Delete、Modify

Miscellaneous : Modify

Monitor : Modify

Namespace : Create、Delete、Modify

Resource : Create、Delete、Modify

Role : Create、Delete、Modify

Service : Modify

Shadows : Create、Delete、Modify
Sharded cluster : Config
SQL admin privilege : Grant、Revoke
SQL privilege : Grant、Revoke
SQL setting : Modify
SQL System Datatype : Create、Delete、Modify
SQL User Datatype : Create、Delete、Modify
SSL configuration : Create、Delete、Modify
Startup setting : Modify
System setting : Modify
User : Create、Delete、Modify
WorkQueue : Create、Delete、Modify

テーブル 8: 処理順

CreateSSLConfigs
ModifySSLConfigs
ModifySystem
DeleteSSLConfigs
ModifyEvent
DeleteEvent
CreateEvent
ModifyMonitor
ModifyJournal
ModifyConfig
ModifyECP
ModifyMiscellaneous
ModifySQL
DeleteWorkQueue
CreateWorkQueue
ModifyWorkQueue
DeleteDevice
DeleteDeviceSubType
CreateDeviceSubType

CreateDevice

ModifyDeviceSubType

ModifyDevice

DeleteMagTapes

CreateMagTapes

ModifyMagTapes

DeleteComPort

CreateComport

ModifyComport

DeleteSqlSysDatatype

CreateSqlSysDatatype

ModifySqlSysDatatype

DeleteSqlUserDatatype

CreateSqlUserDatatype

ModifySqlUserDatatype

DeleteLicenseServer

CreateLicenseServer

ModifyLicenseServer

DeleteMapPackage

DeleteMapGlobal

DeleteMapRoutine

DeleteNamespace

DeleteDatabase

DeleteDatabaseFile

DeleteECPServer

DeleteResource

CreateECPServer

ModifyECPServer

ConfigMirror

CreateResource

CreateDatabaseFile

ModifyDatabaseFile

CreateDatabase

ModifyDatabase

CreateNamespace

ModifyNamespace

CreateMapPackage

CreateMapGlobal

CreateMapRoutine

ModifyMapPackage

ModifyMapGlobal

ModifyMapRoutine

ActivateCPF

ModifyMirrorMember

DeleteMirrors

CreateMirrors

ModifyMirrors

DeleteShadows

CreateShadows

ModifyShadows

ModifyStartup

ModifyService

DeleteUser

DeleteLDAPConfig

DeleteApplication

DeleteDocDB

DeleteRole

CreateRole

CreateDocDB

CreateLDAPConfig

CreateUser

CreateApplication

ModifyResource

ModifyRole

ModifyDocDB

ModifyLDAPConfig

ModifyUser

ModifyApplication

RevokePrivilege

RevokeAdminPrivilege

GrantPrivilege
GrantAdminPrivilege
ConfigShardedCluster

テーブル 9: 対応するクラス別

Config.CPF	ActivateCPF
Config.ComPorts	CreateComPort、DeleteComPort、ModifyComPort
Config.config	ModifyConfig
Config.Devices	CreateDevice、DeleteDevice、ModifyDevice
Config.DeviceSubTypes	CreateDeviceSubType、DeleteDeviceSubType、ModifyDeviceSubType
Config.ECP	ModifyECP
Config.ECPServers	CreateECPServer、DeleteECPServer、ModifyECPServer
Config.Journal	ModifyJournal
Config.LicenseServers	CreateLicenseServer、DeleteLicenseServer、ModifyLicenseServer
Config.MagTapes	CreateMagTapes、DeleteMagTapes、ModifyMagTapes
Config.MapGlobals	CreateMapGlobal、DeleteMapGlobal、ModifyMapGlobal
Config.MapPackages	CreateMapPackage、DeleteMapPackage、ModifyMapPackage
Config.MapRoutines	CreateMapRoutine、DeleteMapRoutine、ModifyMapRoutine
Config.MirrorMember	ModifyMirrorMember
Config.Mirrors	CreateMirror、DeleteMirrors、ModifyMirrors
Config.Miscellaneous	ModifyMiscellaneous
Config.Monitor	ModifyMonitor
Config.Namespaces	CreateNamespace、DeleteNamespace、ModifyNamespace
Config.Shadows	CreateShadows、DeleteShadows、ModifyShadows
Config.SQL	ModifySQL
Config.SqlSysDatatypes	CreateSqlSysDatatype、DeleteSqlSysDatatype、ModifySqlSysDatatype
Config.SqlUserDatatypes	CreateSqlUserDatatype、DeleteSqlUserDatatype、ModifySqlUserDatatype
Config.Startup	ModifyStartup
Config.WorkQueues	CreateWorkQueue、DeleteWorkQueue、ModifyWorkQueue
Security.Applications	CreateApplication、DeleteApplication、ModifyApplication
Security.DocDBs	CreateDocDB、DeleteDocDB、ModifyDocDB
Security.Events	CreateEvent、DeleteEvent、ModifyEvent
Security.LDAPConfigs	CreateLDAPConfig、DeleteLDAPConfig、ModifyLDAPConfig
Security.Resources	CreateResource、DeleteResource、ModifyResource
Security.Roles	CreateRole、DeleteRole、ModifyRole
Security.Services	ModifyService

Security.SSLConfigs	CreateSSLConfigs、ModifySSLConfigs、DeleteSSLConfigs
Security.System	ModifySystem
Security.Users	CreateUser、DeleteUser、ModifyUser
SQL コマンド GRANT、REVOKE	GrantAdminPrivilege、RevokeAdminPrivlege
SYS.Database	CreateDatabase、DeleteDatabaseModifyDatabase、CreateDatabaseFile、DeleteDatabaseFile、ModifyDatabaseFile
%SYSTEM.SQL.Security	GrantPrivilege、RevokePrivilege

テーブル 10: アルファベット順

ActivateCPF
ConfigMirror
ConfigShardedCluster
CreateApplication
CreateComPort
CreateDatabase
CreateDatabaseFile
CreateDevice
CreateDeviceSubType
CreateDocDB
CreateECPServer
CreateEvent
CreateLDAPConfig
CreateLicenseServer
CreateMagTapes
CreateMapGlobal
CreateMapPackage
CreateMapRoutine
CreateMirrors
CreateNamespace
CreateResource
CreateRole

CreateShadows

CreateSqlSysDatatype

CreateSqlUserDatatype

CreateSSLConfigs

CreateUser

CreateWorkQueue

DeleteApplication

DeleteComPort

DeleteDatabase

DeleteDatabaseFile

DeleteDevice

DeleteDeviceSubType

DeleteDocDB

DeleteECPServer

DeleteEvent

DeleteLDAPConfig

DeleteLicenseServer

DeleteMagTapes

DeleteMapGlobal

DeleteMapPackage

DeleteMapRoutine

DeleteMirrors

DeleteNamespace

DeleteResource

DeleteRole

DeleteShadows

DeleteSqlSysDatatype

DeleteSqlUserDatatype

DeleteSSLConfigs

DeleteUser

DeleteWorkQueue

GrantAdminPrivilege

GrantPrivilege

ModifyApplication

ModifyComPort

ModifyConfig

ModifyDatabase

ModifyDatabaseFile

ModifyDevice

ModifyDeviceSubType

ModifyDocDB

ModifyECP

ModifyECPServer

ModifyEvent

ModifyJournal

ModifyLDAPConfig

ModifyLicenseServer

ModifyMagTapes

ModifyMapGlobal

ModifyMapPackage

ModifyMapRoutine

ModifyMirrorMember

ModifyMirrors

ModifyMiscellaneous

ModifyMonitor

ModifyNamespace

ModifyResource

ModifyRole

ModifyService

ModifyShadows

ModifySQL

ModifySqlSysDatatype

ModifySqlUserDatatype

ModifySSLConfigs

ModifyStartup

ModifySystem

ModifyUser

ModifyWorkQueue

RevokeAdminPrivilege

RevokePrivilege