
# コンテナ内でのインターシステムズ製品の実行

Version 2024.1
2024-06-03

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

コンテナ内でのインターシステムズ製品の実行
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03




インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

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

# コンテナ内でのインターシステムズ製品の実行

このドキュメントでは、コンテナを使用するソフトウェア導入の利点を説明し、コンテナに、インターシステムズ提供のイメージを使用して InterSystems IRIS® と InterSystems IRIS ベースのアプリケーションを導入するために必要な情報を提供します。

必要なイメージのダウンロードを含め、InterSystems IRIS コンテナの実行を迅速に開始する方法については、″InterSystems IRIS の基礎：InterSystems IRIS コンテナの実行″ を参照してください。簡単なオンラインの実践練習については、″Deploying and Customizing InterSystems IRIS Containers″ を参照してください。

無料の一時ライセンスが付属する、InterSystems IRIS Community Edition 用のコンテナ・イメージは、InterSystems Container Registry または Docker Hub から入手できます。詳細は、″InterSystems IRIS の導入と操作″ の ″独自のシステムへの InterSystems IRIS Community Edition の導入″ を参照してください。

## 1 コンテナを使用する利点

コンテナは、依存関係がすべて満たされ分離された状態で、プラットフォーム独立の完全にポータブルなランタイム・ソリューションにアプリケーションをパッケージ化するので、次のような利点が生まれます。

- コンテナはコードとデータを明確に区分し、煩雑な処理を完全に分離するので、アプリケーションの導入およびアップグレードが容易になります。
- コンテナはきわめて効率的です。コンテナ内のアプリケーションは、そのアプリケーションを実行し、必要な接続、サービス、インタフェースへのアクセスを可能にするために必要な要素のみとパッケージ化されます。またコンテナは、その他のいずれの実行可能プログラムよりも多くのリソースを必要とすることのない単一オペレーティング・システム プロセスとして実行されます。
- コンテナでは、環境の間でアプリケーションを混乱なく移行することができます（例えば、開発からテストへ、その後でプロダクションへ）。これにより、別々の環境で異なる目標を立てている部門の間でよく生じる競合が少なくなります。開発者は最新のコードとライブラリに集中でき、品質開発者はテストと欠陥の説明に、そして運用エンジニアはネットワーキング、高可用性、データの持続性などの全体的なソリューションのインフラストラクチャに集中できます。
- コンテナは、さまざまな組織がビジネスとテクノロジのニーズにまったく新しい方法で対応するために必要な俊敏性、柔軟性、および再現性を備えています。コンテナでは、アプリケーションのプロビジョニング・プロセス（ビルド・フェーズを含む）が実行プロセスと明確に分離され、DevOps アプローチがサポートされるため、組織はより俊敏性の高い一貫したアプリケーション提供方法とアーキテクチャ（マイクロサービス）を採用できます。

このような利点により、コンテナはアプリケーションの自然な構成要素となり、より簡単、迅速で、高い再現性と強固さを持つアプローチでアプリケーションの提供と導入が推進されます。

インターシステムズのプロダクト・マネージャが提供するコンテナおよびコンテナ・イメージの概要は、InterSystems Developer Community の ″What is a Container?″ および ″What is a Container Image?″ を参照してください。

## 2 Docker コンテナのプラットフォーム

特に、Docker コンテナはあらゆる場所で採用されており、パブリック・クラウドとプライベート・クラウドで使用され、仮想マシン（VM）とベア・メタルでサポートされています。Docker は幅広く浸透しており、大手のパブリック・クラウド ″infrastructure as a service″（IaaS）プロバイダはすべて、企業に役立つようにコンテナ・サービスをサポートしています。企業は Docker

コンテナを使用して、インフラストラクチャの取り扱いはクラウド・プロバイダに任せることによって、システム管理コストを削減できます。

インターシステムズは以前から Docker コンテナ内で InterSystems IRIS をサポートしており、お客様がこの革新的なテクノロジを活用できるように力を注いでいます。

Docker テクノロジに関する技術情報と、最初からステップごとに説明した学習資料については、Docker のドキュメンテーション・サイトを参照してください。

# 3 コンテナ内の InterSystems IRIS

コンテナは、コンテナ化されたアプリケーションを実行するために必要な要素のみをパッケージ化し、アプリケーションをネイティブに実行するので、標準の使い慣れたアプリケーション構成、動作、およびアクセスを提供します。Linux で実行される InterSystems IRIS に習熟している場合、Linux ベースの InterSystems IRIS コンテナが実行する物理、仮想、またはクラウドのシステムやディストリビューションに関係なく、さまざまな Linux システムで実行する従来の InterSystems IRIS インスタンスの場合とまったく同じ方法で、これらのコンテナを操作します。

コンテナ内での InterSystems IRIS の導入および使用に関する詳細は、"InterSystems IRIS コンテナの使用" を参照してください。コンテナ化された InterSystems IRIS のいくつかの重要な機能について、以下のセクションで簡単に説明します。

- インターシステムズ提供のイメージ ー コンテナ・イメージは実行可能パッケージであり、コンテナはイメージの実行時インスタンスです。インターシステムズは、"InterSystems IRIS イメージの使用法" で説明しているように、InterSystems IRIS の完全インストール・インスタンスを含むさまざまなイメージを提供しています。これには、IntegratedML と InterSystems IRIS for Health を備えた InterSystems IRIS 用のイメージのほか、無料の Community Edition インスタンスも含まれます。インターシステムズ提供の InterSystems IRIS イメージを InterSystems IRIS ベースのアプリケーションを含むイメージの基礎として使用することもできます。詳細は、"InterSystems IRIS イメージの作成" を参照してください。
- 安全なコンテナ ー すべての InterSystems IRIS イメージには、非 root インスタンス (root 特権を持たず、すべての所有権を保持するユーザ **irisowner** によってインストールされたインスタンス) が含まれます。したがって、root が所有するものはなく、**irisowner** 以外のユーザが所有または実行するものはありません。非常に厳格なセキュリティの下で InterSystems IRIS を導入する場合、ロック・ダウン・イメージを使用して、非 root であることに加え、ロック・ダウン・セキュリティによってインストールされた、コンテナ化されたインスタンスを導入できます。詳細は、"InterSystems IRIS コンテナのセキュリティ" を参照してください。
- iris-main プログラム ー iris-main プログラムにより、InterSystems IRIS やその他の製品はコンテナ内で実行するアプリケーションの要件を満たすことができます。コンテナの起動時に開始されるメイン・プロセスであるエントリ・ポイント・アプリケーションは、作業が完了するまでブロックされる (つまり待機する) 必要がありますが、InterSystems IRIS を起動するコマンドはブロック・プロセスとしては実行されません。iris-main プログラムは、InterSystems IRIS を起動した後、ブロック・エントリ・ポイント・アプリケーションとして実行を継続することによって、この問題を解決しています。このプログラムは、コンテナ内の InterSystems IRIS の動作の調整を助けるいくつかのオプションも提供します。iris-main の詳細は、"iris-main プログラム" を参照してください。
- 永続的な %SYS 機能 ー コンテナ化されたアプリケーションはホスト環境から分離されているので、永続データを書き込みません。コンテナが削除され、新しいコンテナに置き換えられるときに、アプリケーションがコンテナ内で書き込んだ内容が失われるためです。このため、コンテナ化されたアプリケーションの導入で重要な点は、コンテナの外部にデータを保管し、他のコンテナや後続のコンテナがそのデータを使用できるようにすることです。

  永続的な %SYS 機能を使用すると、インスタンス固有のデータ (ユーザ定義、監査レコード、ログ・ファイル、ジャーナル・ファイル、WIJ ファイルなど) の永続ストレージが可能になり、コンテナ内での InterSystems IRIS の実行時に、単一インスタンスを時間の経過と共に複数のコンテナで順番に実行できるようになります。例えば、永続的な %SYS を使用して InterSystems IRIS コンテナを実行すると、元のコンテナを停止して、元のコンテナで作成されたインスタンス固有のデータを使用する新しいコンテナを実行することによって、インスタンスをアップグレードすることができま

す。アップグレードの詳細は、"InterSystems IRIS コンテナのアップグレード" を参照してください。永続的な %SYS の詳細は、"永続インスタンス・データを保存するための永続的な %SYS" を参照してください。

- コンテナ・イメージのカタログ ― InterSystems IRIS イメージに加え、インターシステムズでは、次のような関連イメージを提供しています。
  - **webgateway** ― InterSystems Web ゲートウェイと Web サーバの両方を導入します。これは、InterSystems IRIS ベースのアプリケーションのコンテナ化された導入のための Web サーバ・コンポーネントを提供するものです。詳細は、"Web ゲートウェイ・コンテナを使用した Web アクセス" を参照してください。
  - **arbiter** ― ミラーリングされた導入の一環としてアービター・ノードを導入します。詳細は、"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。
  - **iris-operator** ― InterSystems Kubernetes Operator (IKO) を導入します。これは、オープンソースの Kubernetes コンテナのオーケストレーション・エンジンを、InterSystems IRIS 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・インスタンスを表す IrisCluster というカスタム・リソースで拡張します。
  - **iam** ― InterSystems API Manager (IAM) を導入します。
  - **sam** ― InterSystems System Alerting and Monitoring (SAM) を導入します。
  - **iscreports_server** ― InterSystems Reports サーバを導入します。
  - **passwordhash** ― プレーン・テキストのパスワードを、InterSystems IRIS PasswordHash 構成パラメータを使用する必要がある、ソルトを付けてハッシュ化したパスワードに変換します。詳細は、"認証とパスワード" を参照してください。

InterSystems Container Registry の説明およびそれに含まれるイメージのリストとダウンロードの詳細は、"InterSystems Container Registry の使用" を参照してください。

# 4 コンテナの基本

ここでは、Docker コンテナの作成と使用に関する重要な基本要素について説明します。

- コンテナの内容
- コンテナ・イメージ
- コンテナの実行

## 4.1 コンテナの内容

基本的に、Docker コンテナは、子プロセスを生成できる単一のプライマリ・プロセスを実行します。単一のブロック・プロセス (作業が完了するまで待機するもの) によって管理できるあらゆる処理を、コンテナにパッケージ化して実行できます。

コンテナ化されたアプリケーションは完全にコンテナ内で維持されますが、コンテナを実行するオペレーティング・システム (OS) 上でアプリケーションが完全に実行されることはなく、アプリケーションを実行するオペレーティング・システム全体がコンテナに収容されることもありません。代わりに、コンテナのアプリケーションは、ホスト・システムのカーネルでネイティブに実行されます。一方で、アプリケーションに必要な要素と、必要な接続、サービス、およびインタフェースへのアクセスを可能にするために必要な要素 (ファイル・システムを含む実行時環境、コード、ライブラリ、環境変数、および構成ファイル) のみが、コンテナで提供されます。

コンテナはこれらの要素のみをパッケージ化し、アプリケーションをネイティブに実行するので、非常に効率的であり (分離した管理可能なオペレーティング・システム・プロセスとして実行され、他の実行可能プログラムより多くのメモリを消費

することはありません)、かつ完全にポータブルです(既定ではホスト環境からの完全な分離を保ち、ローカル・ファイルとポートにアクセスするのはそのように構成された場合のみ)。それと同時に、標準の使い慣れたアプリケーション構成、動作、およびアクセスを提供します。

アプリケーションをホスト環境から分離することは、コンテナ化の非常に重要な要素であり、さまざまな考慮事項があります。おそらく最も重要なことは、明示的に構成されない限り、コンテナ化されたアプリケーションは永続データを書き込まないことです。これは、コンテナが削除されて新しいコンテナに置き換えられると、アプリケーションがコンテナ内で書き込んだ内容が失われるためです。通常はデータの永続性がアプリケーションの要件なので、コンテナの外部にデータを保管し、他のコンテナや後続のコンテナがそのデータを使用できるようにすることが、コンテナ化されたアプリケーションの導入では重要な面です。

## 4.2 コンテナ・イメージ

コンテナ・イメージは実行可能パッケージであり、一方でコンテナはイメージの実行時インスタンスです。つまり、イメージが実際に実行されるときにメモリ内でコンテナになります。この意味では、イメージとコンテナは実行可能形式で存在する他のソフトウェアと同様です。イメージは実行可能プログラムであり、コンテナはイメージの実行結果として作成される実行ソフトウェアです。

Docker イメージは Dockerfile 内で定義され、その最初の形態は、コンテナ内で実行されるすべてのものに対して実行時環境を提供する基本イメージです。例えば、インターシステムズは InterSystems IRIS イメージの基礎として Ubuntu 20.04 LTS を使用しているため、コンテナ内のインターシステムズのイメージから作成された InterSystems IRIS インスタンスは、Ubuntu 20.04 LTS 環境で実行されます。次に、アプリケーションの実行準備に必要なすべての指定が行われます(例えば、ファイルのコピーやダウンロード、環境変数の設定、アプリケーションのインストールなど)。最後のステップでは、アプリケーションの起動が定義されます。

このイメージを作成するには、Dockerfile の場所を指定して docker build コマンドを発行します。作成されたイメージはローカル・ホストのイメージ・レジストリに配置され、ここから他のイメージ・レジストリにコピーできます。

## 4.3 コンテナの実行

コンテナ・イメージを実行してコンテナ (つまり、メモリ内のイメージ・インスタンスとそれを実行するカーネル・プロセス) を作成するには、以下のように 3 つの別々の Docker コマンドを実行する必要があります。

1. docker pull ー レジストリからイメージをダウンロードします。
2. docker create ー コンテナ・インスタンスとそのパラメータを定義します。
3. docker start ー コンテナを開始 (起動) します。

ただし、利便性を考え、これらのコマンドを組み合わせて順番に実行する docker run コマンドが用意されています。これが、コンテナを作成して開始するための通常の手段です。

docker run コマンドにはいくつかのオプションがあり、コンテナ・インスタンスを作成するコマンドが、インスタンスの操作の存続期間にわたる特性を定義することは重要な注意点です。実行中のコンテナは、停止してから再起動できますが(プロダクション環境では通常行われません)、docker run コマンドによって決定された実行の性質は変更できません。例えば、docker run コマンドのオプション (例 : --volume /home/storage:/storage3) を使用して、コンテナ内でボリュームとしてストレージ場所をマウントできますが、コマンドでこの方法によってマウントされたボリュームは、そのイメージのインスタンス化の間は固定され、変更したり追加したりすることはできません。

コンテナ化されたアプリケーションを変更する場合、例えばアップグレードしたり、コンポーネントを追加したりする場合には、既存のコンテナを削除し、docker run コマンドで別のイメージをインスタンス化することによって、新規コンテナを作成して起動します。新規コンテナそのものは前のコンテナに関連付けられていませんが、コンテナを作成して起動するコマンドで同じポートを公開して同じネットワークに接続し、同じ外部ストレージ場所をマウントすると、前のコンテナは実質的に置き換えられます (InterSystems IRIS コンテナのアップグレードの詳細は、"InterSystems IRIS コンテナのアップグレード" を参照してください)。

重要　InterSystems IRIS コンテナに NFS の場所を外部ボリュームとしてマウントすることはサポートされていません。このようにマウントしようとすると、iris-main によって警告が発行されます。

注釈　他の UNIX® や Linux のコマンドと同様に、docker run などの docker コマンドのオプションは長形式（前に 2 つのハイフンを付ける）で指定することも、短形式（前に 1 つ付ける）で指定することもできます。このドキュメントでは、わかりやすさのために全体で長形式を使用します。例えば、-v ではなく --volume を使用します。

# 5 InterSystems IRIS コンテナの使用

このセクションでは、作成したインターシステムズのイメージを使用して InterSystems IRIS コンテナを実行するために必要な手順について説明します。以下のトピックがあります。

- InterSystems IRIS コンテナの自動導入
- InterSystems IRIS イメージの使用法
- iris-main プログラム
- 永続インスタンス・データを保存するための永続的な %SYS
- Web ゲートウェイ・コンテナを使用した Web アクセス
- InterSystems IRIS コンテナの実行
- InterSystems IRIS コンテナのアップグレード
- InterSystems IRIS イメージの作成
- InterSystems IRIS のコンテナ化ツール

## 5.1 InterSystems IRIS コンテナの自動導入

コンテナはさまざまな点で自動導入に適しています。InterSystems IRIS データ・プラットフォームでは、導入後に完全に機能するマルチコンテナ・トポロジ（シャード・クラスタ、分散キャッシュ・クラスタ、ミラーなど）の自動導入方法が 2 つ提供されています。高度なメソッドには以下のようなものがあります。

- InterSystems Kubernetes Operator

  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。導入するコンテナ化されたサービスと、そのサービスを管理するポリシーを定義すると、Kubernetes は、必要なリソースを可能な限り効率的な方法で透過的に提供します。また、導入が指定値から外れた場合は導入を修復またはリストアするほか、拡張を自動またはオンデマンドで行います。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 の使用方法の詳細は、"InterSystems Kubernetes Operator の使用" を参照してください。

- 構成マージ

  Linux および UNIX® システムで利用可能な構成マージ機能を使用すると、宣言型構成マージ・ファイルを導入内の各インスタンスに適用するだけで、同じイメージから導入した InterSystems IRIS コンテナ（または同じキットからインストールしたローカル・インスタンス）の構成を変更することができます。

このマージ・ファイルは、既存のインスタンスに適用することもでき、インスタンスの構成パラメータ・ファイル (CPF) を更新します。CPF にはインスタンスのほとんどの構成設定が含まれており、これらの設定は、インスタンス導入後の最初の起動を含め、開始時に毎回、CPF から読み取られます。導入時に構成マージを適用すると、インスタンスと共に提供された既定の CPF が実質的に独自の更新バージョンに置き換えられます。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、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ます。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。

前のセクションで説明したように、IKO には構成マージ機能が組み込まれています。

構成マージを使用した自動導入のユース・ケースの例を含む、構成マージの使用法の詳細は、"構成マージを使用した InterSystems IRIS の自動構成" を参照してください。

重要　マージ・ファイルを指定する **ISC_CPF_MERGE_FILE** 環境変数を使用して構成マージでコンテナをデプロイすると、そのファイルの更新が継続的に監視されます。コンテナが実行中である限り、更新が発生すると直ちにマージされます。つまり、単にマージ・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ます。詳細は、"構成マージを使用した InterSystems IRIS の自動構成" の "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" を参照してください。

## 5.2 InterSystems IRIS イメージの使用法

以下のセクションでは、インターシステムズ提供の InterSystems IRIS イメージの使用に関する重要な事項について説明します。以下のトピックがあります。

- インターシステムズでサポートされるコンテナ導入プラットフォーム
- Docker のインストール
- InterSystems IRIS イメージのダウンロード
- インターシステムズ提供のイメージの既定値の検出
- InterSystems IRIS コンテナのライセンス・キー
- InterSystems IRIS コンテナのセキュリティ
- InterSystems IRIS コンテナを使用したミラーリング

### 5.2.1 インターシステムズでサポートされるコンテナ導入プラットフォーム

インターシステムズのコンテナ・イメージは Open Container Initiative (OCI) の仕様に準拠しているため、オンプレミスとパブリック・クラウドの両方において、Linux ベースのオペレーティング・システム上の任意の OCI 準拠ランタイム・エンジンでサポートされます。

注釈　このドキュメントで示している具体的な説明や手順は、Linux 上の Docker での使用を対象としています。Docker Desktop (Windows および macOS 向け) では、Linux プラットフォームの場合のようにコンテナをネイティブのプロセスとして実行するのではなく、Linux VM を作成し、その VM をそれぞれのプラットフォーム・バーチャライザ下で実行してコンテナをホストします。このような追加の層によって複雑さが増すため、インターシステムズは現時点では Docker Desktop をサポートしていません。

ただし、インターシステムズは、テストやその他特定の目的のために、インターシステムズ提供の InterSystems IRIS ベースのコンテナを Windows または macOS で実行することが必要な場合があることも理解しています。インターシステムズ提供のコンテナ・イメージと共に使用するために適用した場合の Docker for Windows と Docker for Linux の違いについてインターシステムズが認知しているものに関する詳細は、InterSystems Developer Community の "Using InterSystems IRIS Containers with Docker for Windows" を参照してください。Docker for Windows の使用に関する一般情報については、Dockerドキュメンテーションの "Getting started with Docker for Windows" を参照してください。

### 5.2.2 コンテナ・ランタイム・エンジンのインストール

インターシステムズのコンテナを導入するコンテナ・ランタイム・エンジン (Docker や Podman など) をサーバにインストールします。

### 5.2.3 InterSystems IRIS イメージのダウンロード

https://containers.intersystems.com/ にある InterSystems Container Registry (ICR) には、InterSystems IRIS イメージを含む、インターシステムズから入手可能なすべてのイメージのリポジトリが含まれています。"InterSystems Container Registry の使用" では、ICR から入手できるイメージを紹介すると共に、WRC 認証情報を使用してレジストリに対して認証し、イメージをダウンロードできるようにする方法について説明しています。

注釈　認証を使用せずに、13 か月分の無料の組み込みライセンスが付属した (いくつかの制限を含む) InterSystems IRIS Community Edition イメージを ICR からダウンロードすることもできます。Community Edition イメージは、Docker Hub でも入手できます。詳細は、"InterSystems IRIS の導入と操作 " の "独自のシステムへの InterSystems IRIS Community Edition の導入" を参照してください。

組織のプライベート・イメージ・レジストリで既に InterSystems IRIS イメージが利用可能になっている場合もあります。その場合は、担当の管理者から、レジストリの場所と認証に必要な認証情報を入手してください。ICR または組織のレジストリのどちらかにログインしたら、docker pull コマンドを使用してイメージをダウンロードできます。以下の例は、ICR からのプルを示しています。

```
$ docker login containers.intersystems.com
Username: pmartinez
Password: **********
$ docker pull containers.intersystems.com/intersystems/iris:latest-em
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
$ docker images
REPOSITORY                                      TAG             IMAGE ID      CREATED      SIZE
containers.intersystems.com/intersystems/iris 2023.2.0.299.0 15627fb5cb76  1 minute ago 1.39GB
containers.intersystems.com/intersystems/sam  1.0.0.115       15627fb5cb76  3 days ago   1.33GB
acme/centos                                     7.3.1611        262f7381844c  2 weeks ago  192MB
acme/hello-world                                latest          05a3bd381fc2  2 months ag  1.84kB
```

注釈　このドキュメントで示しているイメージ・タグは、例として紹介しているだけです。InterSystems Container Registry (ICR) に移動して、最新のリポジトリとタグを参照してください。

### 5.2.4 インターシステムズ提供のイメージの既定値の検出

以下の例で示しているように、インターシステムズのコンテナの既定値は、docker inspect コマンドを使用して必要な情報を検出できるように、標準のラベル・メカニズムを通して公開されています。インターシステムズのテクノロジに精通して

いるユーザは、一般的な既定のポートやその他の有用な情報を認識できるはずです(このコマンドの出力形式については、Docker ドキュメントの "Format command and log output" を参照してください)。

```
$ docker inspect -f {{json .Config.Labels}} intersystems/iris:latest-em
  "Labels": {
      "com.intersystems.adhoc-info": "",
      "com.intersystems.platform-version": ":2023.2.0.299.0",
      "com.intersystems.ports.default.arbiter": "2188",
      "com.intersystems.ports.default.license-server": "4002",
      "com.intersystems.ports.default.superserver": "1972",
      "com.intersystems.product-name": "IRIS",
      "com.intersystems.product-platform": "dockerubuntux64",
      "com.intersystems.product-timestamp": "Wed May 16 2022 00:37:59 EST",
      "com.intersystems.product-timestamp.iso8601": "2023-08-16T05:37:59Z",
      "maintainer": "InterSystems Worldwide Response Center <support@intersystems.com>",
      "org.opencontainers.image.created": "2023-08-16T07:57:10Z",
      "org.opencontainers.image.documentation": "https://docs.intersystems.com/",
      "org.opencontainers.image.title": "intersystems/iris",
      "org.opencontainers.image.vendor": "InterSystems",
      "org.opencontainers.image.version": "2023.2.0.299.0"
      }
```

### 5.2.5 InterSystems IRIS コンテナのライセンス・キー

あらゆる InterSystems IRIS インスタンスと同様に、コンテナ内で実行されるインスタンスにもライセンス・キー (通常は iris.key) が必要です。InterSystems IRIS ライセンス・キーの一般情報は、"システム管理ガイド" の "InterSystems IRIS ライセンスの管理" の章を参照してください。

InterSystems IRIS Community Edition イメージ (前のセクションの説明のとおり) には、特別な無料一時ライセンスが付属しています。ただし、通常、ライセンス・キーは、InterSystems IRIS コンテナ・イメージに含まれていません。代わりに、コンテナにアクセスできるストレージ位置 (通常はマウントされたボリューム) にライセンス・キーを置き、コンテナにライセンス・キーをコピーするメカニズムを設定し、コンテナでライセンス・キーを有効にして InterSystems IRIS インスタンスを実行できるようにします。

InterSystems IRIS コンテナのブロック・エントリ・ポイント・アプリケーションとして実行される iris-main プログラムには、ライセンス・キーを扱うためのオプションが用意されています。このオプションは、InterSystems IRIS コンテナを起動する docker start コマンドまたは docker run コマンドで使用できます。--key オプションは、指定した場所から InterSystems IRIS インスタンスの mgr/ ディレクトリにライセンス・キーをコピーします。つまり、ライセンス・キーはインスタンスの起動時に自動的に有効になります。ライセンス・キーをコンテナ内のローカル・ファイル・システムに配置することはできません。通常は、docker run コマンドの --volume オプションでコンテナによってボリュームとしてマウントされるストレージ場所 (多くの場合、永続的な %SYS ボリューム) に配置します。このオプションの構文は、以下のとおりです。

```
--key <key_path>
```

key_path はコンテナ内部からライセンス・キーへのパスです。例えば、ライセンス・キーが license/ というディレクトリに含まれる外部ストレージ場所を /external としてマウントする場合、--key オプションは以下のようになります。

```
--key /external/license/iris.key
```

--key オプションでは、コンテナをアップグレードすることなく、インスタンスのライセンスを更新できます。InterSystems IRIS イメージにより、docker run または docker start コマンドでこのオプションを使用すると、iris-main はステージングされるライセンス・キーの変更を継続的に監視し (元の場所に維持されていると想定)、ファイルの変更が検出されると、現在の mgr/ ディレクトリにコピーされ、%SYSTEM.License.Upgrade() API 呼び出しが行われます。

iris-main のオプションのリストは、"iris-main プログラム" を参照してください。--key オプションの使用例は、"InterSystems IRIS コンテナの実行" を参照してください。

重要　コアベースの IRIS ライセンスを使用している場合、コンテナ・ランタイム・エンジンが、InterSystems IRIS コンテナで使用できるようにするコア数は、ライセンスのコア数以下である必要があります。ホストにそれより多くのコアがある場合は、コンテナの起動時に --cpuset-cpus および --cpus オプションを使用して、コンテナが使用する CPU を制限する必要があります。InterSystems IRIS Community Edition インスタンスが関与する例では、無料のライセンスは 20 コアに制限されます。"独自のシステムへの InterSystems IRIS Community Edition の導入" を参照してください。

### 5.2.6 InterSystems IRIS コンテナのセキュリティ

インターシステムズ提供の InterSystems IRIS イメージを操作する場合、以下に示したような、イメージのセキュリティ関連のメカニズムとオプションを理解することが重要です。

- 所有権とディレクトリ
- 認証とパスワード
- InterSystems IRIS ロック・ダウン・コンテナ

重要　InterSystems IRIS のセキュリティは包括的で、インターシステムズが提供するコンテナ・イメージは非常に厳格なセキュリティ要件をサポートするように設計されていますが、コンテナ化されたインスタンスから見れば問題のあるようなコンテナ化されていないインスタンスに向けて、いくつかの重要なセキュリティ手法が考案されています。特に、人的介入を必要とする操作は、コンテナ化された自動導入および操作には適していません。例えば、インタラクティブにキーを有効化する起動（データベース暗号化キーがアクティブ化されるインスタンスの既定の動作）は、キーの位置のプロンプトで起動が中断され、オペレータはこれに応答する必要があります。代行承認を使用するためのインスタンスの構成など、その他のタスクは、一般にインタラクティブな ^SECURITY ルーチンを使用して実行されます。

構成マージ機能は、任意の目的の構成でコンテナ化されたインスタンスを自動導入できるため、一部のケースでは役立ちますが、セキュリティ・タスクによっては、コンテナ化されたソフトウェア向けのセキュリティ・メカニズムを利用した Kubernetes などのコンテナ・オーケストレータ機能を使用することが最適な対処となる場合もあります。例えばシークレットは、幅広く使用されているメカニズムで、少量の機密情報がオブジェクトに配置され、必要に応じてプラットフォームにより提供されるため、アプリケーション・コードに機密データを含める必要がなくなります。これは、人的介入を必要としない無人のキー有効化による起動を実現するように構成された、コンテナ化された InterSystems IRIS インスタンスに、暗号化キーを提供するための適切で安全な手段です。さらに、Hashicorp の Vault などのサードパーティ・ツールは、Kubernetes などのプラットフォームで機能し、ポリシーや幅広いニーズに合わせた複数のオプションにより機密データを管理および保護するための追加サポートを提供します。

注釈　Web ゲートウェイ・コンテナのセキュリティに関する重要な情報は、"Web ゲートウェイ・コンテナのセキュリティ" を参照してください。

#### 所有権とディレクトリ

インターシステムズ提供のイメージから作成されたコンテナ内の InterSystems IRIS インスタンスは、常に **IRIS** という名前が付けられ、非 root です。これは、root 特権のないユーザ・アカウント **irisowner** (UID 51773) によってインストールおよび所有されていることを意味します。インスタンスを構成するすべてのファイル・システム・エンティティは、**irisowner** によって所有され、root が所有するものはありません。インスタンスではオペレーティング・システム・ベースの認証が有効になっています。つまり、**irisowner** プロセスまたは別のシステムで認証されたクライアントは認証なしでインスタンスに接続できます。コンテナ外から docker exec を使用して発行されたコマンドは、コンテナ内で **irisowner** として実行されるため、このコマンドを使用することで、認証なしで簡単にインスタンスに接続することができます。例えば、**iris273** というコンテナ内のインスタンスの InterSystems ターミナルを認証情報を要求されずに開くには、次のコマンドを使用します。

```
docker exec -it iris273 iris terminal IRIS
```

コンテナ内のインストール・ディレクトリ (mgr/ サブディレクトリを含む) は、/usr/irissys/ です。作業ディレクトリ (iris-main.log などのファイルを含む) は /home/irisowner/、レジストリ・ディレクトリは /home/irisowner/irissys/ です。

これらの構成の詳細を指定するために使用されるインストール・パラメータの詳細は、"必須の環境変数" を参照してください。こういったインストール関連の一般的なトピックの詳細は、"UNIX®、Linux、および macOS でのインストール" を参照してください。

インターシステムズには、作成する InterSystems IRIS ベースのイメージのこういった構成の詳細を決定するためのツールが用意されています。これらのツールについては、"InterSystems IRIS のコンテナ化ツール" を参照してください。

重要　永続的な %SYS 機能を使用してコンテナ化された InterSystems IRIS インスタンスに永続ストレージを提供する場合、このためにマウントおよび指定されるホストのファイル・システムの場所は、ユーザ 51773 によって書き込み可能である必要があります (これを有効にするには、多くの場合、root 特権が必要です)。

Pod Manager ツール (podman) で導入された InterSystems IRIS コンテナで永続的な %SYS を使用する場合は、以下を行う必要があります。

- コンテナを起動する前に、以下のコマンドを発行します。

  ```
  podman unshare chown 51773:51773 $INSTANCEDIR
  ```

  $INSTANCEDIR は、永続的な %SYS ディレクトリを含むホスト・ファイル・システムの場所です。

- Red Hat Security-Enhanced Linux (SELinux) がアクティブな場合は、コンテナの作成時に --privileged=true フラグを含めます。

Kubernetes で InterSystems Kubernetes Operator なしで永続的な %SYS を使用する場合、ポッド指定に次のセキュリティ・コンテキスト設定を含める必要があります。

```
securityContext:
  fsGroup: 51773
```

永続的な %SYS を使用して InterSystems IRIS コンテナをバージョン 2021.2 以降にアップグレードする前に、既存の永続ディレクトリをユーザ 51773 によって書き込み可能にする必要があります。

注釈　**irisowner** ユーザ・アカウントがコンテナをホストするシステムの /etc/passwd ファイルに定義されていない場合、これは、そのシステムの UID (51773) で表されます。

### 認証とパスワード

OS ベースの認証 ("認証ガイド" の "オペレーティング・システム・ベースの認証について" を参照) が、インターシステムズ提供のイメージから作成された、コンテナ内の InterSystems IRIS インスタンスに対して有効化され、パスワード認証が所有者 (**irisowner**) に対して無効化されます。イタリック

InterSystems IRIS は、**_SYSTEM** アカウントを含む複数の事前定義ユーザ・アカウントと共にインストールされます ("認証ガイド" の "事前定義のユーザ・アカウント" を参照)。事前定義のアカウントの既定のパスワードは、**SYS** です。効果的なセキュリティを確保するために、コンテナの導入直後にこの既定のパスワードを変更することが重要です。この変更は、以下のいずれかの方法を使用して実行できます。これらの方法のいずれかを自動導入に組み込むことができます。

注意　既定のパスワードを変更するためにここにリストされた方法のいずれも使用しない場合は、できるだけ速やかに、事前定義のアカウントそれぞれにログインしてパスワードを変更するか、アカウントを無効にする必要があります。

以下に示すように、多くの場合、サーバ・アクセス・プロファイルで InterSystems IRIS インスタンスに対して認証するために使用されるアカウントとしてインターシステムズの Web ゲートウェイに構成される、事前定義した **CSPSystem** アカウントのパスワード (**SYS**) を変更できるのは、iris-main --password-file オプションのみです。特に既定のパスワードを変更するために --password-file オプションを使用しない場合は、導入後にできるだけ早く **CSPSystem** アカウントにログインしてパスワードを変更する必要があります。Web ゲートウェイ認証を構成する方法の詳細は、"Web ゲートウェイ・コンテナのセキュリティ" を参照してください。

重要　インスタンスを導入し、新しい既定のパスワードで実行したら、事前定義のアカウントそれぞれにログインする必要があります。これらのアカウントは、最初のログイン時にパスワードを変更するよう構成されています。これにより、1 つのパスワードを共有するのではなく、選択した新しい個々のパスワードですべてのアカウントのセキュリティが確保されます。代替手段として、これらのアカウントの 1 つ以上を無効にすることもできます。

注釈　既定の設定を使用した場合に生じる、InterSystems IRIS イメージを作成してから 90 日後のパスワードの有効期限切れを回避するため、コンテナ化されたインスタンスは、インスタンス所有者および事前定義アカウントのパスワードが有効期限切れにならないように構成されます。

- PasswordHash CPF 設定

  UNIX または Linux プラットフォームに InterSystems IRIS を自動的に導入する際、最初にインスタンスを起動する前にそれらのインスタンスの既定のパスワードを変更できます。それには、構成パラメータ・ファイル (CPF) の PasswordHash 設定を構成マージ機能と組み合わせて使用します。

  アカウントごとにプレーン・テキストのパスワードを記録するのではなく (セキュリティ・リスクがある)、InterSystems IRIS では、パスワードの不可逆暗号化ハッシュのみが格納されます。ユーザのログイン時に、入力されたパスワードの値が、同じアルゴリズムを使用してハッシュ化され、2 つのバージョンの比較によってユーザが認証されます。格納されるパスワード・ハッシュの詳細は、"承認ガイド" の "インスタンス認証" を参照してください。イタリック

  新たに導入されたインスタンスの事前定義アカウント (1 つ以上のロールが割り当てられている有効なユーザ・アカウント) すべてに対し、格納されたパスワード・ハッシュ (および結果的にパスワード) を設定または変更できます。これには CPF の [Startup] セクションにある PasswordHash 設定を使用します。導入時にこの設定が構成マージ・ファイルに含まれている場合 (UNIX® および Linux システムの場合のみ)、インスタンスの事前定義アカウント (ロールが割り当てられていない **CSPSystem** を除く) の既定のパスワードがカスタマイズされて、**SYS** が、値をハッシュとして指定したパスワードで置き換えられます。

  前述のとおり、導入後直ちに、事前定義アカウントのパスワードを PasswordHash によって設定された新しい既定のパスワードから変更する必要があります。PasswordHash パラメータは特定のインスタンスに対して機能するのは一度だけであるため、インスタンスの CPF に残っていても何の影響も与えません。

  PasswordHash パラメータの引数は、ハッシュ化されたパスワードとそのソルトで、必要に応じて、パスワードのハッシュ化に使用されるハッシュ化アルゴリズムと作業係数も指定できます (ハッシュ化アルゴリズムと作業係数の既定値はそれぞれ SHA512 と 10000 です)。これらすべての引数を以下の例に示します。

  ```
  [Startup]
  PasswordHash=0fadb1a665e04efb5fe929de8457466883e0a3a42c1a34aeec49dbc1fb8c4c40f184658ce180c10898db836,db874dc346d2367ed1b49db9f48baae82b962,10000,SHA512
  ```

  プレーン・テキストのパスワードをこれらの値に変換するために使用するアルゴリズムの説明については、"インスタンス認証" を参照してください。アルゴリズムを適用するためのツールは、%SYSTEM.Encryption API に用意されています。ただし、この変換を手動手順として実行することはお勧めしません。エラーが発生しやすいうえ、時間がかかる可能性が高いためです。利便性のために、InterSystems Container Registry ("InterSystems IRIS イメージのダウンロード" を参照) で、この変換をユーザの代わりに実行して、PasswordHash パラメータのコンテキストに結果を表示するナノコンテナ向けのイメージ **passwordhash** が提供されています。オプションで、使用する作業係数とアルゴリズムを指定できます。指定しない場合は、既定値が使用されます。以下は、このコンテナの使用例です。

  ```
  $ docker run --rm -it containers.intersystems.com/intersystems/passwordhash:1.1 -algorithm SHA512
   -workfactor 10000
  Enter password:
  Enter password again:
  PasswordHash=0fadb1a665e04efb5fe929de8457466883e0a3a42c1a34aeec49dbc1fb8c4c40f184658ce180c10898db836,db874dc346d2367ed1b49db9f48baae82b962,10000,SHA512
  ```

  次に、出力のコピーと貼り付けを実行し、前述のとおり、構成マージ・ファイルの [Startup] セクションに配置します。導入後、事前定義アカウント (**CSPSystem** を除く) の既定のパスワードは、プロンプトで入力したパスワードになります。

注釈　以下に示すように、iris-main --help オプションを使用して使用状況の情報を表示できます。

```
$ docker run containers.intersystems.com/intersystems/passwordhash:1.1 --help
Usage of /passwordhash:
  -algorithm string
        Pseudorandom function to use (default "SHA512")
  -workfactor int
        PBKDF2 Work Factor (default 10000)
```

以下のように、コマンドへの入力としてハッシュ化されるパスワードを指定することもできます。

```
$ echo **** | docker run --rm -i containers.intersystems.com/intersystems/passwordhash:1.1
```

重要　PasswordHash プロパティを使用できるのは、特定の InterSystems IRIS インスタンスに対して一度のみで、なおかつどの事前定義アカウントに対しても既定のパスワードが変更されていない場合のみです。導入後に既定のパスワードを変更しないままにすることが可能であるということは重大なセキュリティ・リスクであるため、構成マージ操作では PasswordHash 設定を使用して導入時に (それ以降ではなく) 既定のパスワードを変更する必要があります (個々のユーザのパスワードを変更する方法は、"承認ガイド" の "既存のユーザ・アカウントの編集" を参照してください)。

注釈　PasswordHash 設定では、空白のパスワードを使用することはできません。

- iris-main --password-file オプション

  iris-main エントリポイント・アプリケーションに対するこのオプションは、コンテナでの最初の起動時に、**CSPSystem** アカウントを含む InterSystems IRIS インスタンスの事前定義のアカウントの既定のパスワードを、ユーザが指定したファイルの内容に変更した後、そのパスワード・ファイルを削除して、再度実行されることがないようにする標識ファイルを作成し、コンテナが起動するたびにこのオプションが呼び出されないようにします。詳細な手順は次のとおりです。

  - 指定したパスワード・ファイルが含まれるディレクトリに標識ファイルが存在する場合、このスクリプトはパスワードの変更を試みずに終了します。
  - 標識ファイルが存在しない場合、スクリプトでは以下が実行されます。
    1. 指定したファイルから新しいパスワードを読み取ります。
    2. インスタンスが実行中の場合はこれをシャットダウンします。
    3. 1 つ以上のロールが割り当てられている有効なユーザ・アカウントすべてのパスワードを変更する API 呼び出しを行い、インスタンスの既定のパスワードを実質的に変更します。
    4. パスワードの変更が正常に完了したら、事前定義の **CSPSystem** アカウントのパスワードを変更する別の API 呼び出しを行います (このセクションで前述のとおり)。
    5. パスワード・ファイルが書き込み可能な場合、スクリプトでは以下が実行されます。
       - パスワード・ファイルを削除します。
       - 標識ファイルを作成します。

       パスワード・ファイルが読み取り専用の場合、標識ファイルは作成されません。これによって、Docker シークレット、Kubernetes シークレット、および同様のテクノロジとの互換性が保たれます。

       iris-main --password-file オプションは、スクリプト changePassword.sh を呼び出します。これは、Linux プラットフォームの InterSystems IRIS のインストール・ディレクトリの下の dev/Container/ にあります (InterSystems が提供する **iris** コンテナ内に含まれます)。独自のツールに統合するため、このスクリプトを別の方法で呼び出すことができます。

--password-file オプションの詳細は、"iris-main プログラム" を参照してください。

- SYS.Container API

  InterSystems IRIS は、API 呼び出し SYS.Container.ChangePassword() を使用して配布されますが、この API 呼び出しは、スクリプトやその他の自動化にも有用です。SYS.Container.ChangePassword() によって、インスタンスの、1 つ以上のロールが割り当てられている有効なユーザ・アカウントすべてのパスワードが、ユーザ指定のファイルのコンテンツに変更されます (Docker シークレット、Kubernetes シークレット、および同様のテクノロジとの互換性のために、読み取り専用パスワード・ファイルを指定するオプションも用意されています)。変更は、インスタンスの最初の起動時に、ログインが可能になる前に実行され、**changePassword.sh** スクリプトによって、さらに結果的に iris-main --password-file オプションによって呼び出されます。これを使用する場合は、長時間パスワードをファイルにコミットすることのリスクに留意してください。

  この API には、SYS.Container.ChangeGatewayMgrPassword() 呼び出しも含まれます (スクリプトからも呼び出される)。これは InterSystems IRIS インスタンスとローカルの Web ゲートウェイ (存在する場合) の両方で、**CSPSystem** アカウントのパスワードを変更します。

  SYS.Container API の詳細は、"SYS.Container API" を参照してください。

### InterSystems IRIS ロック・ダウン・コンテナ

非常に厳格なセキュリティ要件をサポートするため、インターシステムズでは **iris-lockeddown** というイメージを提供しています。このイメージから、安全性の高い InterSystems IRIS コンテナを導入できます。ここでは、このイメージからのコンテナと標準の **iris** イメージからのコンテナ間の差異について詳細に説明します。

注釈　**Iris-lockeddown** イメージの特性は、ベスト・プラクティスの進化と共に変わる可能性があります。現在のセキュリティ手法とお客様が使用されているプロダクション・コンテナ・オーケストレーションの要件を最大限把握したうえで、機能を追加、削除、または変更する可能性があります。

- InterSystems IRIS ロック・ダウン・コンテナ内のインスタンスは、標準の InterSystems IRIS コンテナ内インスタンスの通常のセキュリティ・インストールとは異なり、ロック・ダウン・セキュリティによってインストールされています。ロック・ダウン・セキュリティと通常のセキュリティとの差異の詳細は、"インスタンスの保護" の "インターシステムズのセキュリティのための準備" を参照してください。
- InterSystems System Alerting and Monitoring (SAM) がインスタンスと共に導入されている場合、/api/monitor Web アプリケーションの **[許可された認証方法]** 設定を **[パスワード]** から **[認証なし]** に変更することによって、SAM からインスタンスにアクセスできるようにする必要があります。このためには、管理ポータルの [ウェブ・アプリケーション] ページ (**[システム管理]→[セキュリティ]→[アプリケーション]→[ウェブ・アプリケーション]**) で、左側の列にある [/api/monitor] をクリックして [ウェブ・アプリケーションの編集] ページを表示し、**[セキュリティの設定]** セクションで必要な変更を行って、**[保存]** をクリックします。
- 次のセクションにリストされている標準コンテナ内で定義された環境変数に加え、ロック・ダウン・コンテナ内で **SYS_CONTAINER_LOCKEDDOWN** 変数が **1** と定義されます。

## 5.2.7 InterSystems IRIS コンテナを使用したミラーリング

"高可用性ガイド" の "ミラーリングの構成" で説明している手順に従って、コンテナに導入された InterSystems IRIS インスタンスをキットから導入されたインスタンスと同じ方法でミラーとして構成できます。ただし、以下に挙げるいくつかの点に留意してください。

- (インターシステムズが強く推奨するように) ミラーに対して構成するアービターは、インターシステムズが提供する **arbiter** イメージから導入できます。arbiter イメージは、InterSystems IRIS イメージについての説明と同じ手順を使用してダウンロードできます ("InterSystems IRIS コンテナのセキュリティ" で説明したとおり、これは非 root イメージであることに注意してください)。コンテナ化されていない InterSystems IRIS インスタンスまたはキットからインストールされたアービターも使用できます。

- InterSystems IRIS コンテナおよびアービター・コンテナを導入する場合、"高可用性ガイド" の "ミラー・メンバのネットワーク・アドレス" で説明しているように、ミラー・メンバ、それらの ISCAgent、およびアービターが使用するポートを確実に公開する必要があります。
- 各フェイルオーバーおよび DR 非同期ミラー・メンバの ISCAgent は、InterSystems IRIS が起動する前に自動的に起動するように構成される必要があります。これは、InterSystems IRIS イメージの既定の動作です。InterSystems IRIS コンテナの実行時は、次の iris-main ISCAgent オプションを使用できます。

  --ISCAgent true|false

  `true` の場合、コンテナが既定の ISCAgent ポート 2188 で起動されると、ISCAgent が起動します (これが、オプションが省略された場合の既定の動作です)。`false` の場合、ISCAgent は起動せず、`--ISCAgentPort` オプションは無視されます。

  --ISCAgentPort NNNN

  ポートを指定して、ISCAgent を起動します (オプションが省略された場合の既定値は 2188 です)。このオプションは、`--ISCAgent true` と共に使用できます。指定された値が有効なポート番号ではない場合 (整数でない場合など)、または示されたポートが使用されている場合、ISCAgent は開始に失敗します。
- "高可用性ガイド" の "ミラーリング通信"、"ミラーリング・アーキテクチャとネットワーク構成のサンプル"、および "ミラーの可用性を最適化するためのアービターの配置" を参照して、導入において必要なネットワーキングに関する考慮事項すべてに対処していることを確認してください。

## 5.3 iris-main プログラム

コンテナ内でアプリケーションを実行するために、アプリケーションが満たす必要がある要件がいくつかあります。iris-main プログラムは、InterSystems IRIS とその他の関連製品がこれらの要件を満たすことができるように、インターシステムズによって開発されたものです。

docker run コマンドによって実行されるメインのプロセス (エントリ・ポイントと呼ばれる) は、作業が完了するまでブロック (つまり、待機) する必要があります。長時間実行されるエントリ・ポイント・アプリケーションの場合、このプロセスは意図的に停止されるまでブロックする必要があります。

InterSystems IRIS は通常、iris start コマンドを使用して起動されます。このコマンドは、いくつかの InterSystems IRIS プロセスを生成して、コマンド行に制御を戻します。iris コマンドはブロック・プロセスとして実行されないので、Docker エントリ・ポイント・アプリケーションとしての使用には適していません。

iris-main プログラムは、InterSystems IRIS を起動した後、ブロック・エントリ・ポイント・アプリケーションとして実行を継続することによって、この問題を解決しています。このプログラムはまた、コンテナが停止すると InterSystems IRIS を適切にシャットダウンし、いくつかの有用なオプションを備えています。

このプログラムを使用するには、iris-main バイナリを Dockerfile に追加して、エントリ・ポイント・アプリケーションとして指定します。以下に例を示します。

```
ADD host_path/iris-main /iris-main
ENTRYPOINT ["/iris-main"]
```

重要　iris-main プログラムは、InterSystems IRIS を起動する前に、InterSystems IRIS コンテナで必要となる特定の Linux 機能 (CAP_SETUID や CAP_SETGID など) を確認します。コンテナの実行時環境において、常に必要な機能は一般的に既定で付与されます。一部の環境でのみ必要となるいくつかの機能は付与されない可能性があります。存在しない機能がある場合、iris-main はエラーをログに記録し、インスタンスを起動することなく終了します。この機能チェックは、iris-main オプション --check-caps false を含めることで無効にできます。

Docker は、以下の追加要件をエントリ・ポイント・アプリケーションに適用します。

- docker stop による適切なシャットダウン

Docker は、docker stop コマンドへの応答として、メインのコンテナ・プロセスがシャットダウンすることを予期します。

docker stop の既定の動作では、SIGTERM シグナルをエントリ・ポイント・アプリケーションに送信し、10 秒間待機した後、SIGKILL シグナルを送信します。Kubernetes も同様の方法で動作します。iris-main プログラムは、SIGTERM および SIGINT を傍受し、インスタンスの適切なシャットダウンを実行します。

重要　docker stop コマンドの発行時にインスタンスが特にビジー状態である場合は、完全な停止状態になるまでに 10 秒では十分でない場合があり、その結果 Docker から SIGKILL が送信される可能性があります。SIGKILL はトラップしたり処理したりできません。プログラムが中断されてデータが失われる可能性があるという点で、SIGKILL はマシンの電源を切ることと似ています。SIGKILL を受け取ると、InterSystems IRIS コンテナは次回の起動時に InterSystems IRIS の通常のクラッシュ・リカバリ手順に従います。これを防止するには、docker stop コマンドで --time オプションを使用するか、Kubernetes 構成で terminationGracePeriodSeconds の値を使用して、待機時間を 10 秒より長く設定します。

- docker start による適切な起動

  docker stop コマンド以外の手段によってコンテナが停止した場合 (例えば、Docker デーモンの再起動、またはホストのリブート)、エントリ・ポイント・アプリケーションは docker start コマンドへの応答として、コンテナを再び起動して安定した実行状態にするために必要なタスクをすべて実行する必要があります。このドキュメントの作成時点で、iris-main は不適切に停止した InterSystems IRIS インスタンスに対して特別な処理を行わず、代わりに既存の InterSystems IRIS の機能に依存します。ただし、--before オプションと --after オプションを使用して指定されたすべての操作を実行します (以下の表を参照)。

- docker logs によって取得される標準出力へのログ

  docker logs コマンドが出力を表示できるように、エントリ・ポイント・アプリケーションでコンテナの標準出力に出力を送信することが Docker では期待されます。iris-main プログラムは既定でこれに従い、InterSystems IRIS のログの内容をすべて標準出力に送信します。必要に応じて、-log オプションを使用して、コンテナ内の別のファイルの出力 (例えば、アプリケーションのログ) をコンテナの出力に送信できます。以下に例を示します。

  ```
  docker run iris --log /myapp/logs/myapp.log
  ```

  致命的なエラーが発生すると、iris-main によってメッセージ・ログが示され ("監視ガイド" の "install-dir/mgr ディレクトリのログ・ファイル" を参照)、エラーに関する詳細な情報を確認できます。

  注釈　iris-main プログラムは、そのログ出力を前の出力に追加するように構成されます。これにより、InterSystems IRIS コンテナが再起動された場合に、シャットダウンの方法と理由に関するレコードを引き続き利用できるようにします。

前述の問題に対処するほかに、iris-main はコンテナ内の InterSystems IRIS の動作を調整するために役立ついくつかのオプションを用意しています。iris-main が提供するオプションを以下のリストに示します。これらの使用例は、"InterSystems IRIS コンテナの実行" に記載されています。

iris-main のオプションは、docker run コマンド内でイメージ名の後に指定され、一方で Docker のオプションはその前に指定されます。docker コマンドと同様に、オプションには 2 つのハイフンを使用する長形式と、1 つのみ使用する短形式があります。

| オプション | 説明 | 既定値 |
|---|---|---|
| -i instance,<br>--instance instance | 起動または停止する InterSystems IRIS インスタンスの名前を設定します。(インターシステムズにより配布されるコンテナ内のインスタンスは常に、**IRIS** と名付けられます。) | IRIS |
| -d true\|false、<br>--down true\|false | コンテナのシャットダウン時に (iris stop を使用して) InterSystems IRIS を停止します。 | true |

| オプション | 説明 | 既定値 |
|---|---|---|
| -u true\|false、<br>--up true\|false | コンテナの起動時に (iris start を使用して) InterSystems IRIS を起動します。 | true |
| -s true\|false、<br>--nostu true\|false | シングル・ユーザ・アクセス・モードで InterSystems IRIS を起動します。 | false |
| -k key_file、<br>--key key_file | 指定した InterSystems IRIS ライセンスキー ("InterSystems IRIS コンテナのライセンス・キー" を参照) を、インストール・ディレクトリの mgr/ サブディレクトリにコピーします。 | なし |
| -l log_file,<br>--log log_file | docker logs コマンドを使用したモニタリングのために、標準出力にリダイレクトするログ・ファイルを指定します。 | なし |
| -b command,<br>--before command | InterSystems IRIS を起動する前に実行する実行可能プログラム (シェル・スクリプトなど) を設定します。 | なし |
| -a command,<br>--after command | InterSystems IRIS を起動した後に実行する実行可能プログラムを設定します。 | なし |
| -e command,<br>--exit command | InterSystems IRIS を停止した後に実行する実行可能プログラムを設定します。 | なし |
| -c command、<br>--create command | 他の引数を処理する前にカスタム・シェル・コマンドを実行します。 | なし |
| -t command、<br>--terminate command | 他の引数を処理した後にカスタム・シェル・コマンドを実行します。 | なし |
| -p password_file、<br>--password-file<br>password_file | 事前定義の InterSystems IRIS アカウントの既定のパスワードをファイルに含まれる文字列に変更した後、そのファイルを削除します。<br>重要 "認証とパスワード" で説明されているように、このオプションはスクリプトやその他の自動化で有用です。これを使用する場合は、長時間パスワードをファイルにコミットすることのリスクに留意してください。既定のパスワードが変更されている場合でも、コンテナの起動後の、各事前定義アカウントへの最初の手動ログインには、必須の既定パスワード変更が含まれます。 | なし |
| --ISCAgent true\|false | コンテナの起動時に、既定の ISCAgent ポート 2188 で ISCAgent を起動します。false の場合、ISCAgent は起動せず、--ISCAgentPort オプションは無視されます。 | true |
| --ISCAgentPort NNNN | ポートを指定して、ISCAgent を起動します。--ISCAgent true と共に使用できます。 | 2188 |
| --check-caps false | InterSystems IRIS の起動前の自動 Linux 機能チェックを無効にします。 | true |
| --version | iris-main のバージョンを出力します。 | 該当なし |

| オプション | 説明 | 既定値 |
|---|---|---|
| -h、<br>--help | 使用法の情報を表示して終了します。 | 該当なし |

## 5.4 永続インスタンス・データを保存するための永続的な %SYS

このセクションでは、InterSystems IRIS の永続的な %SYS 機能について説明し、この機能の使用法を説明します。この機能により、InterSystems IRIS がコンテナ内で実行されているときに、インスタンス固有のデータおよびユーザ作成データベースの永続ストレージが可能になります。

### 5.4.1 InterSystems IRIS の永続的な %SYS の概要

コードとデータの分離は、コンテナ化の主な利点の 1 つです。実行中のコンテナは、適切なデータ・ソースを処理できる"純粋なコード"を表します。ただし、アプリケーションとプログラムはすべて、動作と履歴のデータ (例えば構成と言語の設定、ユーザ・レコード、ログ・ファイルなど) を生成し、維持します。これらは、1 つのコンテナの存続期間を超えて保持され、アップグレードを可能にする必要があります。このため、コンテナ化は一般に、永続的なデータ・ストレージでアプリケーション・データベースを含めプログラム関連データを保持するニーズに対処する必要があります。これには、保持するデータと永続的な場所を特定し、アップグレード後、または前のコンテナで障害が発生した後で、新しいコンテナをこの場所に送信できるメカニズムが必要です。

永続的な %SYS 機能は、必要なインスタンス固有データ (ログ、ジャーナル、WIJ ファイル、および **IRISSYS** などのシステム・データベースなど) を、コンテナのホストのファイル・システム上で選択した場所に複製し、その新しい (複製された) 場所でデータを使用するためインスタンスをリセットすることにより、InterSystems IRIS コンテナでこれを実現します。永続的な %SYS を使用するには、選択した場所をコンテナ内のボリュームとしてマウントし、コンテナの起動時に指定される環境変数でこれを識別する必要があります。InterSystems IRIS インスタンスはコンテナ化されたままですが、そのインスタンス固有データは、アプリケーション・データが保存されるデータベースと同様にコンテナの外部に存在しており、コンテナとインスタンスの再起動後にも永続し、インスタンスのアップグレードに利用できるようになっています (アップグレードの詳細は、"InterSystems IRIS コンテナのアップグレード" を参照してください)。

重要　コードとデータの分離を維持するには、InterSystems IRIS から、またはiris merge コマンドを使用して、マウントされた外部ボリューム上にすべての InterSystems IRIS データベースを作成し、インターシステムズが提供するイメージから InterSystems IRIS イメージを構築する ("InterSystems IRIS イメージの作成" を参照) 際にこれらを追加して、これらのデータベースが永続的な %SYS によって永続的なストレージに複製されるようにすることをお勧めします。

### 5.4.2 永続的な %SYS ディレクトリの内容

永続的な %SYS ディレクトリは、コンテナが最初に起動されるときに作成された状態では、InterSystems IRIS インストール・ツリーのサブセットを含みます。内容は以下のようなものですが、これらに限られるわけではありません。

- iris.cpf という名前の構成パラメータ・ファイル (CPF)。他の InterSystems IRIS インスタンスの場合と同様に、ファイルの追加バージョン (旧バージョンと _LastGood_.cpf) が作成されます。("構成マージを使用した InterSystems IRIS の自動構成"、"構成パラメータ・ファイル・リファレンス")
- Web ゲートウェイ構成ファイルとログ・ファイルを含む /csp ディレクトリ。(Web ゲートウェイ・ガイド)

  注釈　Web ゲートウェイ・コンテナには同様の永続ストレージ機能があり、その機能を使用した場合 ("Web ゲートウェイ・コンテナの実行オプション" を参照)、このディレクトリは独自の永続ストレージ上に配置されます。

- インスタンスのバージョン (2023.2.0.299.0 など) を含むファイル /dist/install/misc/buildver。

- インスタンスの Web サーバの構成ファイルである /httpd/httpd.conf ファイル。(このリリース用のオンライン・ドキュメント "インターシステムズのサポート対象プラットフォーム" の "サポート対象テクノロジ" の章にある "サポート対象の Web サーバ")
- 以下の内容を含む /mgr ディレクトリ。
  - IRISSYS システム・データベース (IRIS.DAT ファイルと iris.lck ファイル、およびストリーム・ディレクトリを含む)、および iristemp、irisaudit、iris、および user の各ディレクトリ (IRISTEMP、IRISAUDIT、IRIS、および USER の各システム・データベースを含む)。(システム提供データベースおよび IRISSYS データベースおよびカスタム項目)
  - ライト・イメージ・ジャーナリング・ファイル IRIS.WIJ (ファイル・システムの分離を実現するために配置が変更される場合があります)。("データ整合性ガイド" の "ライト・イメージ・ジャーナリングとリカバリ" の章)
  - ジャーナル・ファイルを含む /journal ディレクトリ (ファイル・システムの分離を実現するために配置が変更される場合があります)。("データ整合性ガイド" の "ジャーナリング" の章。このドキュメントの "コンテナ化された InterSystems IRIS のファイル・システムの分離" も参照)
  - 一時ファイル用の /temp ディレクトリ。
  - messages.log、journal.log、SystemMonitor.log などのログ・ファイル。その他のログが初期に存在する場合があり、いくつかは必要に応じて作成されます (例えば、バックアップおよびミラーのジャーナル・ログ)。(" 監視ガイド" の "InterSystems IRIS ログの監視")

    注釈　永続的な %SYS のアクティビティは、messages.log ファイルに記録されます。この機能の使用中に問題が生じた場合は、このログを調べると役に立つ情報が得られることがあります。コンテナの外部からこのログを読み取る方法については、"iris-main プログラム" を参照してください。コンテナ化された InterSystems IRIS インスタンスの管理ポータルへのアクセスの詳細は、"Web ゲートウェイ・コンテナを使用した Web アクセス" を参照してください。

  - InterSystems IRIS ライセンス・キー・ファイル iris.key (InterSystems IRIS イメージに含まれている場合はコンテナの起動時、またはコンテナの実行中にライセンスが有効化されたとき)。("システム管理ガイド" の "InterSystems IRIS ライセンスの管理" の章にある "ライセンス・キーの有効化")
  - いくつかの InterSystems IRIS システム・ファイル。

- 上の最初の箇条書きでリストされている標準の InterSystems IRIS データベース以外の、インスタンスで定義された読み取り専用ではないすべてのデータベース。これは、インターシステムズが提供するイメージをベースにしたユーザ作成イメージ ("InterSystems IRIS イメージの作成" を参照) 内のインスタンスに追加されたデータベースが永続的なデータに含まれていることを確認するものです。データベース・ディレクトリがコンテナのインストール・ディレクトリの下にある場合 (/usr/irissys/mgr の下など)、永続的な %SYS ディレクトリ内の対応する場所にコピーされます。データベース・ディレクトリがインストール・ディレクトリの下にない場合は、インストール・ディレクトリに新しいフォルダ db が作成され、ここにこれらがコピーされます。

### 5.4.3 永続的な %SYS ディレクトリの場所

このシステムに不可欠なインスタンス固有の情報が保管される場所を選択する際には、以下の考慮事項に留意してください。

- 適切なバックアップとリストアの手順が利用可能であること ("データ整合性ガイド" の "バックアップとリストア" の章)。
- 導入している高可用性メカニズム ("高可用性ガイド")。
- 使用可能なストレージ・スペースと拡張の余地 ("システム管理ガイド" の "InterSystems IRIS の管理" の章にある "ローカル・データベースの管理")。

永続的な %SYS ディレクトリを初期化するには、指定されたボリューム上で少なくとも 200 MB の使用可能なスペースが必要です。ただし、ジャーナル・レコードなどの処理用ファイルや、システム・データベースの拡張など、さまざまな理由からディレクトリ内のデータの量が大幅に増える可能性があります。

マウント・ボリュームの最上位ではなく、サブディレクトリを永続的な %SYS の場所として指定することをお勧めします。例えば、外部ファイル・システムの場所がコンテナ内のボリューム /external としてマウントされている場合、/external を永続的な %SYS の場所として指定しないでください。代わりに、/external/durable などの /external 上のディレクトリを指定してください。

重要

"InterSystems IRIS コンテナのセキュリティ" で説明したとおり、**iris** および **iris-lockeddown** イメージ内のインスタンスは、ユーザ **irisowner** (UID 51773) によってインストールおよび所有されているため、永続的な %SYS に指定するファイル・システムの場所は、最初に **irisowner** によって書き込み可能な状態にする必要があります。例えば、次のコマンドを使用してこれを行います。

```
chown 51773:51773 root-of-durable-%SYS-directory
```

これを有効にするには、多くの場合、root 特権が必要です。

Kubernetes で InterSystems Kubernetes Operator なしで永続的な %SYS を使用する場合、ポッド指定に次のセキュリティ・コンテキスト設定を含める必要があります。

```
securityContext:
  fsGroup: 51773
```

### 5.4.4 永続的な %SYS を使用した InterSystems IRIS コンテナの実行

永続的な %SYS を使用するには、以下のオプションを docker run コマンドに組み入れます。

```
--volume /volume-path-on-host:/volume-name-in-container
--env ISC_DATA_DIRECTORY=/volume-name-in-container/durable-directory
```

ここで、volume-path-on-host はコンテナによってマウントされる永続的なストレージ場所のパス名、volume-name-in-container はコンテナ内でのマウント・ボリュームの名前、durable_directory はそのマウント・ボリュームに作成する永続的な %SYS ディレクトリの名前です。例えば、以下のように指定します。

```
docker run --detach
  --volume /data/dur:/dur
  --env ISC_DATA_DIRECTORY=/dur/iconfig
  --name iris21 intersystems/iris:latest-em
```

この例では、永続的な %SYS ディレクトリはコンテナ外部の /data/dur/iconfig と、コンテナ内部の /dur/iconfig です。

重要

前の例で説明したように、プロダクション・システムで InterSystems IRIS コンテナに外部ボリュームをマウントする場合は、バインド・マウントを使用することを強くお勧めします。ただし、デモや、Linux 以外のプラットフォームに移植できるようにしたいものをテストしたり作成したりする場合など、状況によっては、ディレクトリ・パスや許可などに関連した問題を排除できるため、名前付きボリュームを使用することをお勧めします。各メソッドの詳細は、Docker ドキュメントの "Manage data in Docker" を参照してください。

InterSystems IRIS コンテナに NFS の場所を外部ボリュームとしてマウントすることはサポートされていません。このようにマウントしようとすると、iris-main によって警告が発行されます。指定された永続的な %SYS の場所が、ファイル・システムのタイプが **cifs** であるか、文字列 **fuse** を含むいずれかのタイプであるマウント・ボリューム上にある場合も、同様の警告が発行されます。

注釈　--publish オプションは、InterSystems IRIS インスタンスのスーパーサーバ・ポート (1972) をホストに公開し、外部エンティティ (他のコンテナ内のエンティティを含む) がそのインスタンスに接続できるようにします。Docker TCP スタックに関する潜在的な問題を回避するために、--publish オプションを --net host オプションに置き換えます。これによりコンテナからホスト・ネットワーク層に既定のソケットが公開されます。--net host オプションは、実行している InterSystems IRIS コンテナがホスト上で唯一の InterSystems IRIS コンテナになる場合、よりシンプルで高速な方法になります。ただし、--publish オプションでは、ホストで公開するコンテナ・ポートをより詳細に制御できるため、安全性はこちらの方が高いと考えられます。

これらのオプションを使用して InterSystems IRIS コンテナを実行する際には、以下の処理が行われます。

- 指定された外部ボリュームがマウントされます。
- コンテナ内部の InterSystems IRIS インストール・ディレクトリが読み取り専用に設定されます。
- 前述の例の ISC_DATA_DIRECTORY 環境変数 iconfig/ によって指定された永続的な %SYS ディレクトリが既に存在し、/mgr サブディレクトリを含んでいる場合は、インスタンスの内部ポインタはすべてそのディレクトリにリセットされ、インスタンスはそのディレクトリに含まれているデータを使用します。データの InterSystems IRIS バージョンがインスタンスのバージョンと一致しない場合は、アップグレードされたものと想定され、データは必要に応じてインスタンスのバージョンにアップグレードされます (アップグレードについては、"InterSystems IRIS コンテナのアップグレード" を参照してください)。
- ISC_DATA_DIRECTORY によって指定されている永続的な %SYS ディレクトリが存在しないか、存在していて空の場合は、以下の処理が行われます。
  - 必要に応じて、指定された永続的な %SYS ディレクトリが作成されます。
  - "永続的な %SYS ディレクトリの内容" に示されているディレクトリとファイルが、インストールされた場所から永続的な %SYS ディレクトリにコピーされます (オリジナルは元の場所に残ります)。
  - インスタンスの内部ポインタはすべて永続的な %SYS ディレクトリにリセットされ、インスタンスはそのディレクトリに含まれているデータを使用します。

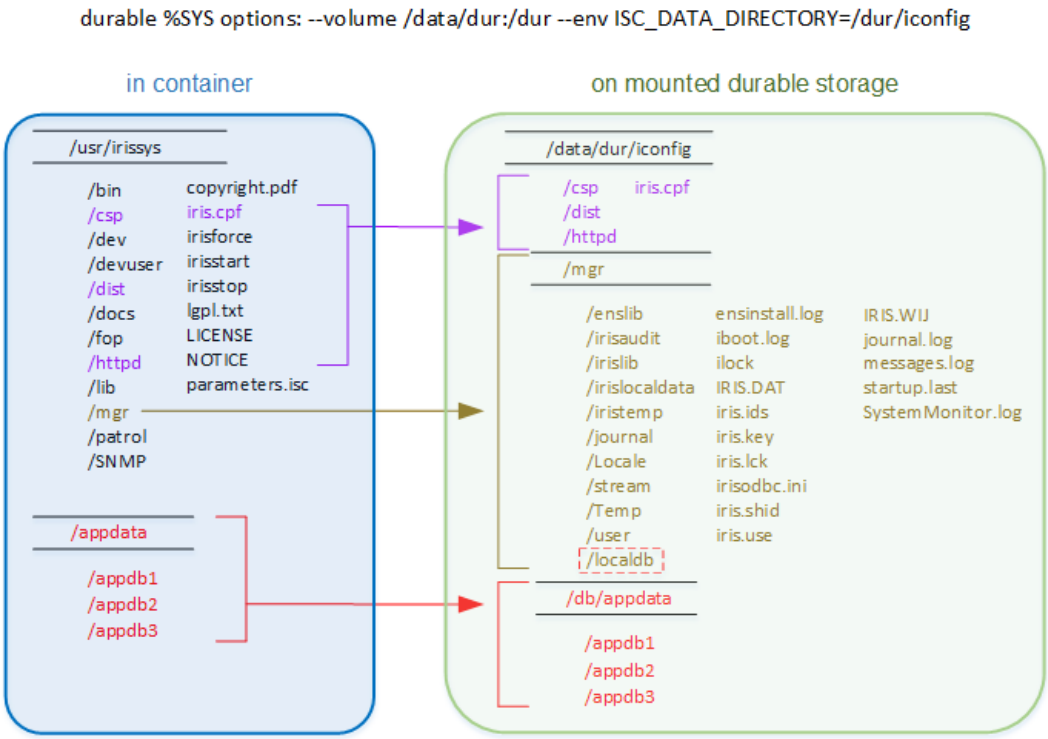
  何らかの理由で、永続的な %SYS データのコピーと内部ポインタのリセットのプロセスが失敗した場合、永続的な %SYS ディレクトリは「不完全」としてマークされます。同じディレクトリに対してもう一度操作を実行した場合、そのディレクトリ内のデータは、永続的な %SYS プロセスが開始する前に削除されます。
- ISC_DATA_DIRECTORY 環境変数によって指定されている永続的な %SYS ディレクトリが既に存在していてデータ (ファイルまたはサブディレクトリ) が含まれていても、/mgr サブディレクトリが含まれていない場合、データはコピーされません。コンテナは起動せず、"iris-main プログラム" で説明されているように、iris-main によってその理由 (永続的な %SYS 以外のデータがディレクトリ内に存在する) が標準出力に記録されます。

ここに示した例の場合、コンテナ **iris21** 内で実行される InterSystems IRIS インスタンスは、"永続的な %SYS ディレクトリの内容" に示されているファイルすべての永続ストレージ用のディレクトリとして、ホスト・パス /data/dur/iconfig (コンテナ内でのパスは /dur/iconfig) を使用するように構成されています。永続的な %SYS がホスト・ディレクトリ /data/dur/iconfig (コンテナ・ディレクトリ /dur/iconfig) にまだ存在しない場合は、インストール・ディレクトリからここにコピーされます。どちらの場合も、インスタンスの内部ポインタはコンテナ・ディレクトリ /dur/iconfig に設定されます。

永続的な %SYS を含む InterSystems IRIS コンテナを起動するさまざまな例については、"InterSystems IRIS コンテナの実行" を参照してください。

次の図は、コンテナ内と永続的な %SYS により複製されたマウントされた外部ストレージ (示したオプションで指定したとおり) の、InterSystems IRIS のインストール・ディレクトリおよびユーザ・データベースの関係を示しています。インストール・ディレクトリ外の 3 つのアプリケーション・データベースは、/data/dur/iconfig/db に複製されている一方、インストール・ディレクトリ内の /mgr の下の **LOCALDB** データベースは、同じ場所 (/data/dur/iconfig/mgr/localdb) に複製されています。

図 1: 永続的な %SYS により複製されたファイル・システム・オブジェクト




## 5.4.5 永続的な %SYS ディレクトリの場所の確認

永続的な %SYS ディレクトリの場所を手動で確認する場合、またはこの場所をプログラムによって受け渡す場合は、以下のように 3 つのオプションがあります。

- コンテナ内でシェルを開き (例えば、docker exec -it container_name bash を使用して)、以下のいずれかを行います。

```
echo $ISC_DATA_DIRECTORY
iris list
```

  注釈　　iris コマンドについての詳細は、"InterSystems IRIS インスタンスの制御" を参照してください。

- InterSystems IRIS 内で、$SYSTEM.Util.InstallDirectory() または $SYSTEM.Util.GetEnviron('ISC_DATA_DIRECTORY') を呼び出します。

## 5.4.6 永続的な %SYS が指定され、マウントされていることの確認

ISC_DATA_DIRECTORY 環境変数を使用してコンテナが実行される場合は、指定されたボリュームが正常にマウントされた場合のみ、ポインタが永続的な %SYS ファイルに設定されます。

- ISC_DATA_DIRECTORY が指定されても、必要な --volume /external_host:/durable_storage オプションが docker run コマンドから省略されている場合、インスタンスは起動に失敗し、エラー・メッセージが生成されます。
- --volume オプションが含まれていても、Docker が指定ボリュームを正常にマウントできない場合、外部ストレージ・ディレクトリとボリュームがコンテナ内に作成されます。この場合、インスタンス・データが永続的な %SYS ディレクトリにコピーされます ("永続的な %SYS を使用した InterSystems IRIS コンテナの実行" の "ISC_DATA_DIRECTORY によって指定されている永続的な %SYS ディレクトリが存在しない場合" で説明されています)。

ISC_DATA_DIRECTORY が指定されていない場合、InterSystems IRIS インスタンスはコンテナ内のインスタンス固有のデータを使用するので、前のインスタンスのアップグレードとしては動作できません。

このため、永続的な %SYS を使用するには、InterSystems IRIS コンテナを実行するために使用するすべてのメソッドに、これら 2 つのオプションが組み込まれていることを確認する必要があります。

### 5.4.7 コンテナ化された InterSystems IRIS のファイル・システムの分離

パフォーマンスと復元可能性の両方を高めるために、各 InterSystems IRIS インスタンスのプライマリおよびセカンダリのジャーナル・ディレクトリは 2 つの分離したファイル・システムに配置することをお勧めします。これらは、InterSystems IRIS の実行可能プログラム、インスタンスのシステム・データベース、および IRIS.WIJ ファイルをホストするファイル・システムとも別にしてください (オプションで、最後のものは 4 つ目のファイル・システムに配置します)。ただし、InterSystems IRIS のインストール後、プライマリとセカンダリのジャーナル・ディレクトリは同じパス (install-dir/mgr/journal) に設定されるので、永続的な %SYS の使用時は、両方のディレクトリが永続的な %SYS ディレクトリ内の /mgr/journal に設定される可能性があります。

"構成マージを使用した InterSystems IRIS の自動構成" で説明しているように、構成マージ・ファイルを使用して、導入に個別のファイル・システムを構成することもできます。コンテナの起動後、管理ポータル (後続のセクションを参照) を使用するか、CPF (iris.cpf) を編集することにより、プライマリ・ディレクトリとセカンダリ・ディレクトリの外部の場所を再構成できます (InterSystems IRIS インスタンスをアップグレードする新規イメージを実行する際に、再配置先のボリュームを必ず指定する限り)。

注釈　永続的な %SYS ディレクトリの使用時は、IRIS.WIJ ファイルと一部のシステム・データベースは既に InterSystems IRIS の実行可能プログラム (コンテナ内部にある) から分離されています。状況によっては、IRIS.WIJ ファイルとアプリケーション・データベースを代わりに同じ場所に配置すると、パフォーマンスが向上する場合があります。

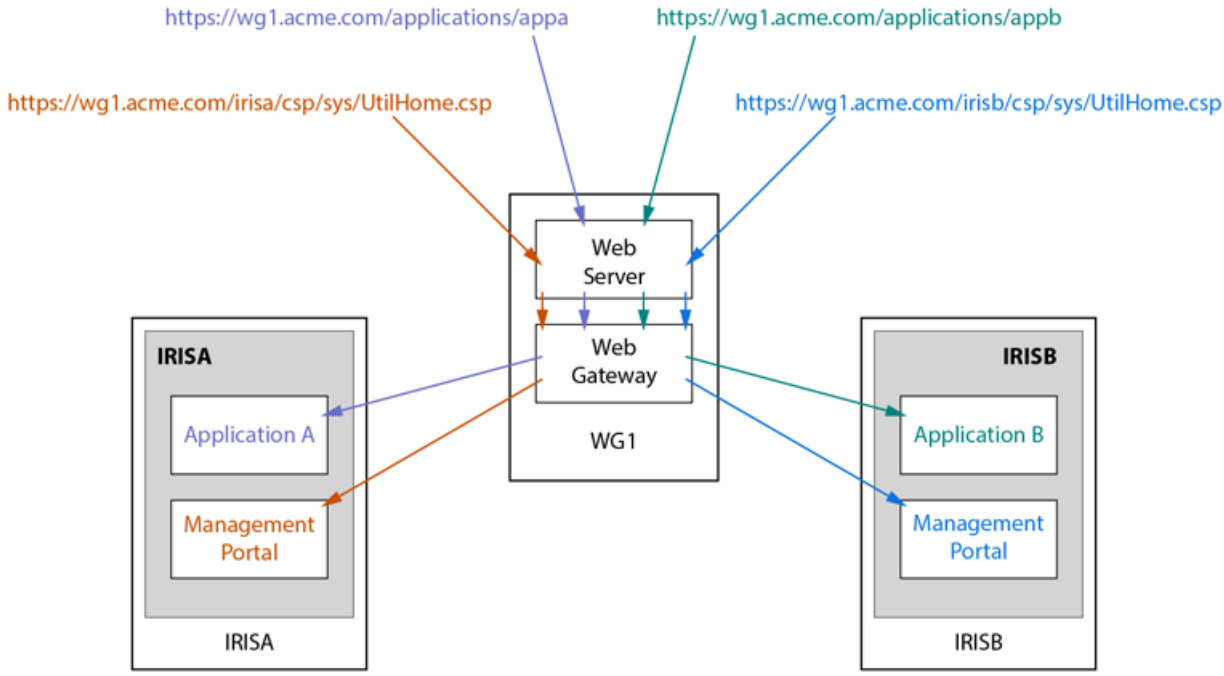
InterSystems IRIS のファイル・システムの分離に関する詳細は、"ファイル・システムの分離" を参照してください。

## 5.5 Web ゲートウェイ・コンテナを使用した Web アクセス

InterSystems IRIS Web ゲートウェイは、任意の Web アプリケーションに、ホスト Web サーバと InterSystems IRIS の間の通信層を提供し、HTTP/HTTPS 要求を InterSystems IRIS に渡し、要求元に渡される応答を返します。インターシステムズが提供する **webgateway** イメージには、Web ゲートウェイと Web サーバの両方が含まれており、スタンドアロンの InterSystems IRIS コンテナでコンテナ化された InterSystems IRIS クラスタに Web サーバ・コンポーネントを提供します。

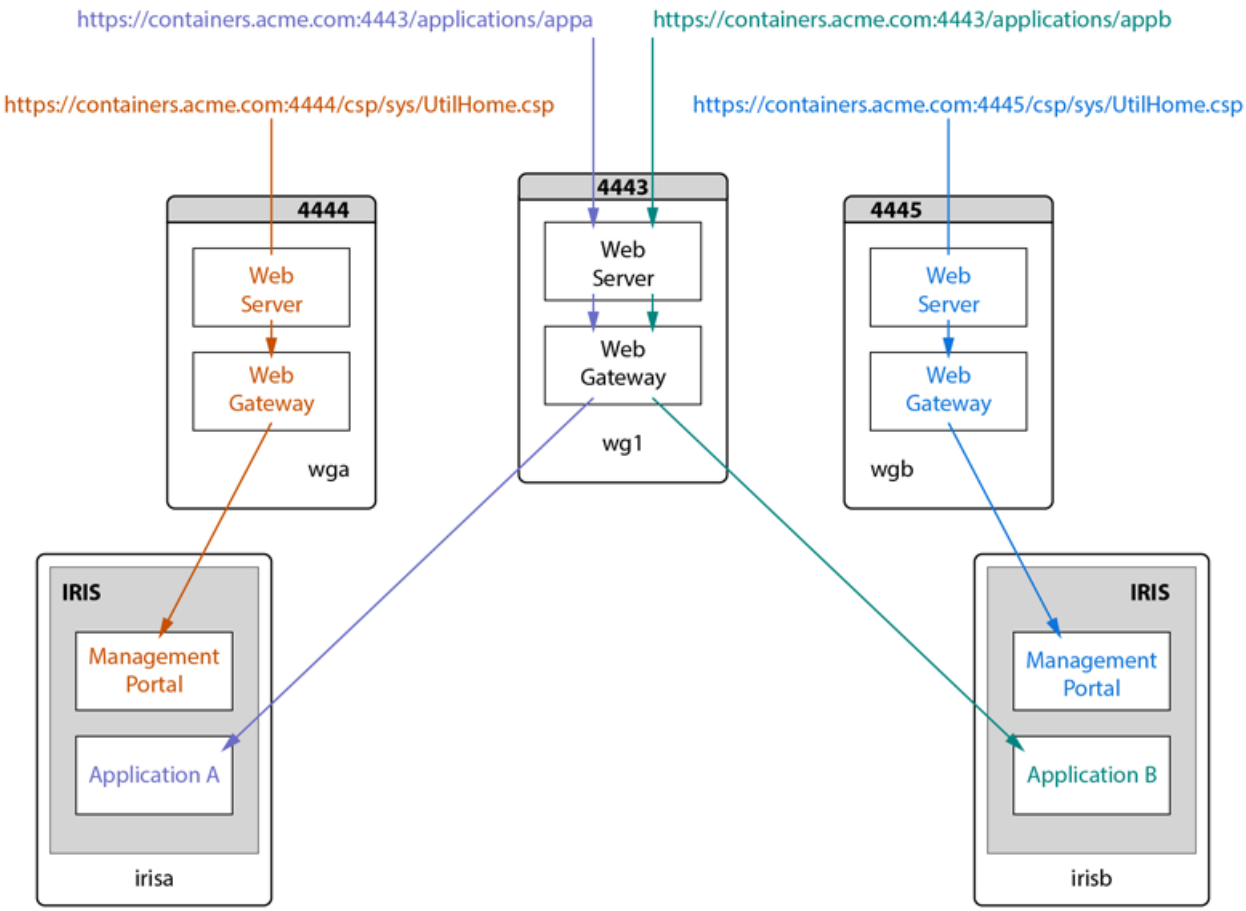
Web ゲートウェイでは、アプリケーション接続と共に、InterSystems IRIS インスタンスの Web ベースの管理ポータル、および InterSystems IRIS に組み込まれているその他の Web アプリケーションに対する要求が処理されます。"Web ゲートウェイ・ガイド" の "InterSystems Web アプリケーション URL の構造" で説明しているように、インスタンス固有の文字列を管理ポータルの基本の URL に挿入することによって、単一の Web ゲートウェイから、ローカルにインストールされている複数のインスタンスに管理ポータル要求を送信できるようになるため、アプリケーション接続と管理ポータルの両方の要求を複数の InterSystems IRIS インスタンスに送信することが可能になります。これを以下の図で示します。

図 2: アプリケーション接続と管理ポータルの要求を複数の InterSystems IRIS インスタンスに送信する Web ゲートウェイ




ただし、この機能は、コンテナ化されたインスタンスでは使用できません。そのため、InterSystems IRIS コンテナそれぞれに、管理ポータル (およびその他の Web アプリケーション) へのアクセス専用の Web ゲートウェイ・コンテナが必要です。1 つの InterSystems IRIS コンテナのみが導入されている場合、単一の Web ゲートウェイ・コンテナはアプリケーション接続と管理ポータルの両方の要求をそのコンテナに送信できますが、複数インスタンスがコンテナ化されている導入では、アプリケーション接続を分散するために 1 つ以上の追加の Web ゲートウェイが必要です。これを、単一ホスト上の複数コンテナの導入を表す以下の図に示します。ここでは、コンテナの作成時に標準の HTTPS ポート 443 に対して公開されたホスト・ポートを含めることによって、URL は適切な Web ゲートウェイ・コンテナに向けられます。

図 3: 専用の Web ゲートウェイとアプリケーション用 Web ゲートウェイを備えた InterSystems IRIS コンテナ

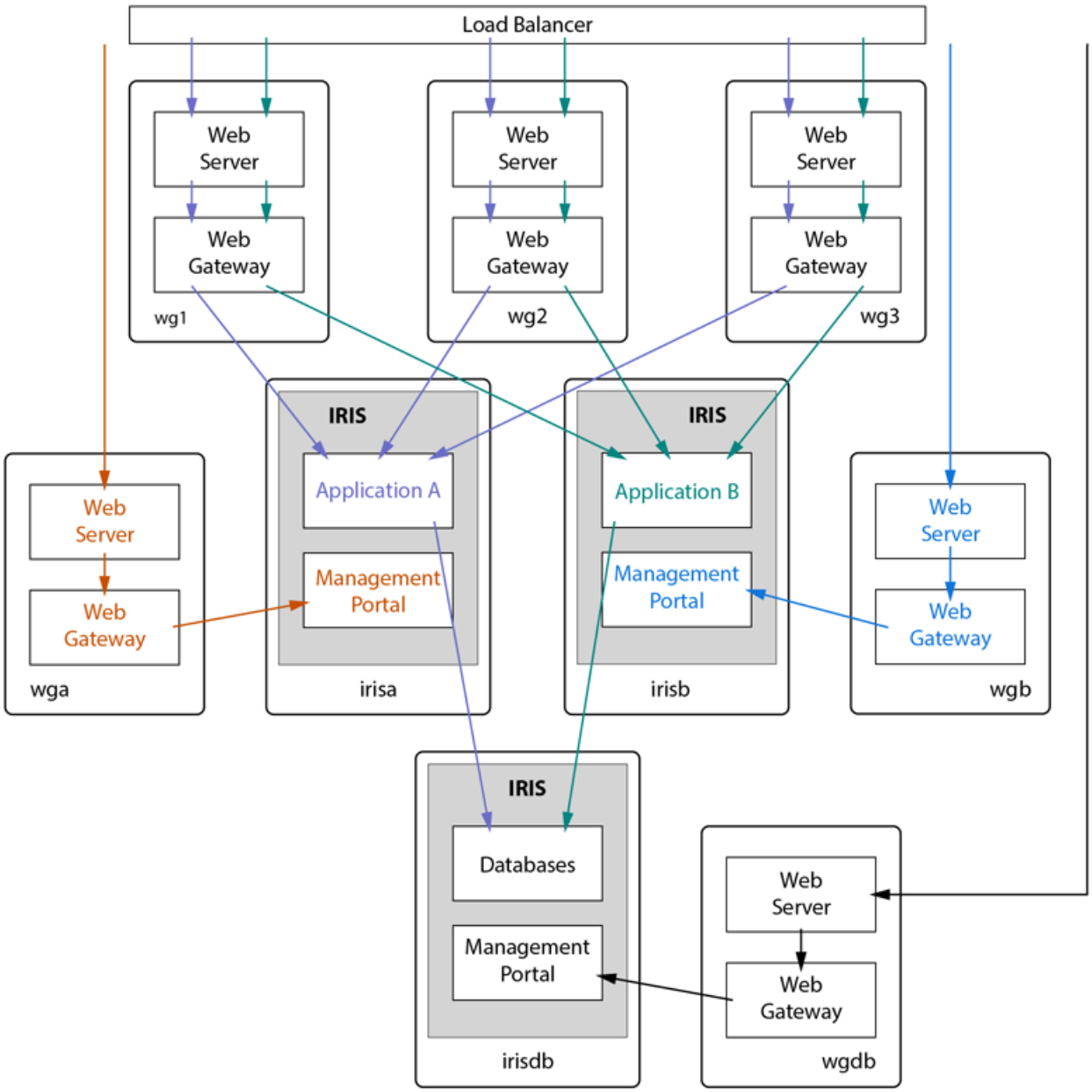



注釈　Docker Compose を使用して、スタンドアロンの InterSystems IRIS コンテナを専用の Web ゲートウェイ・コンテナと共に、必要なすべてのファイルといくつかの有用なスクリプトも含めて導入する簡単な例を https://github.com/intersystems-community/webgateway-examples/tree/master/demo-compose で確認できます。

IKO (″InterSystems IRIS コンテナの自動導入″ を参照) では、管理ポータルへのアクセス専用の Web ゲートウェイ・コンテナの導入は、各 InterSystems IRIS ポッドにサイドカー・コンテナとして追加することで実行されます。詳細は、″InterSystems Kubernetes Operator の使用″ を参照してください。

Web ゲートウェイが通信できる InterSystems IRIS インスタンスは、Web ゲートウェイのサーバ・アクセス・プロファイルで定義されます。コンテナの場合、これらのプロファイルには、コンテナの作成時に各インスタンスのスーパーサーバ・ポートに対して公開されるホスト・ポートを含める必要があります。要求は Web ゲートウェイに構成されているアプリケーション・アクセス・プロファイルに基づいてこれらのインスタンスに送信されます。これらのプロファイルは、Web サーバで構成されているアプリケーション・パスに一致します。Web ゲートウェイ・ノードが、複数のインスタンスにわたってアプリケーション接続が分散されている Web サーバ層で使用される場合、アプリケーション・プロファイルによって、どの接続がどのインスタンスに割り当てられるのかが決定されます。例えば、シャード・クラスタのベスト・プラクティスは、クラスタ内のすべてのデータ・ノードにわたって均等にアプリケーション接続を分散させることです。このような場合には、サーバ・プロファイルに定義されているすべてのデータ・ノード・コンテナにわたって均等にアプリケーション接続が分散されるようにアプリケーション・プロファイルを構成します。一方、分散キャッシュ・クラスタでは、さまざまなアプリケーションに対する要求をさまざまなアプリケーション・サーバに転送することによってキャッシュ効率を最大化できます。このような場合、接続を不均等に分散するようにアプリケーション・パスを構成します。これを以下の図で示します。

図 4: 不均等に分散されるアプリケーション接続と専用の Web ゲートウェイを備えた、コンテナ化された分散キャッシュ・クラスタ




注釈　複数の InterSystems IRIS インスタンスにわたってアプリケーション接続を分散する Web サーバ層の負荷分散に関する重要な説明は、"Web ゲートウェイ・ガイド" の "負荷分散、フェイルオーバー、ミラー構成" を参照してください。

重要　以前のバージョンの InterSystems IRIS では、Web ゲートウェイおよび事前構成されたプライベート Web サーバは、コンテナを含め、既定で InterSystems IRIS と共にインストールされていました。このため、バージョン 2023.2 にアップグレードする場合は、すべての導入スクリプトおよびツールを更新して、このドキュメントで説明されている新しい導入オプションが反映されるようにする必要があります。また、2023.2 InterSystems IRIS コンテナは、InterSystems Kubernetes Operator 3.6 以降でのみ使用する必要があります。

InterSystems IRIS のテストと評価を行うユーザの利便性のために、InterSystems IRIS Community Edition イメージには、引き続き Web ゲートウェイと事前構成された Web サーバが含まれています。この Web サーバは、コンテナ化されたインスタンスの Web サーバのポート 52773 に対して公開されているホスト・ポートであればどこでも (例えば、管理ポータルにアクセスする場合など) アクセスできます。

### 5.5.1 インターシステムズの Web ゲートウェイ・イメージ

インターシステムズが提供する **webgateway** イメージは 3 種類あり、それぞれがコンテナ化された導入に Web サーバ・コンポーネントを提供します。

- **webgateway** イメージには、次のコンポーネントが含まれ、コンテナ内の示された場所に root によってインストールされます。
  - /opt/webgateway 内のインターシステムズの Web ゲートウェイ。
  - /etc/apache2 内の Apache Web サーバ。
- **webgateway-nginx** イメージには、次のコンポーネントが含まれ、root によってインストールされます。
  - /opt/webgateway 内のインターシステムズの Web ゲートウェイ。
  - /opt/nginx 内の Nginx Web サーバ。
- **webgateway-lockeddown** イメージは、非常に厳格な要件を満たすように設計されており、**irisowner** (UID 51773) によってインストール、所有、および実行される、次の非 root コンポーネントが含まれます。
  - /home/irisowner/webgateway にロックダウン・セキュリティでインストールされた、インターシステムズの Web ゲートウェイ。
  - /home/irisowner/apache にインストールされ、標準ポート 80 ではなく、ポート 52773 を使用するように構成された Apache Web サーバ。

注釈　非 root インストールとロックダウン・セキュリティの詳細は、"InterSystems IRIS コンテナのセキュリティ" を参照してください。

### 5.5.2 Web ゲートウェイの構成

Web ゲートウェイの構成には Web ゲートウェイ管理ページが使用されますが、構成は CSP.ini ファイルに格納されます (ただし、InterSystems IRIS インスタンスの構成は iris.cpf ファイルに格納されます)。Web ゲートウェイは、いくつかの基本設定を含む既定の最小バージョンの CSP.ini ファイルと共にインストールされます。Web ゲートウェイ・コンテナの場合のみ、Web ゲートウェイとやり取りするように Web サーバを構成する CSP.conf ファイルが、Web サーバの構成に追加されます。3 つのすべての **webgateway** イメージには、既定の最小バージョンの CSP.ini ファイルと Apache または Nginx 固有の CSP.conf ファイルの両方が、前述のリストで示したディレクトリに含まれています。ただし、独自のバージョンのファイルまたは両方のファイルを提供してこれらを上書きすることもできます。(CSP.ini ファイルの内容の詳細は、"Web ゲートウェイ・ガイド" の "Web ゲートウェイ構成ファイル (CSP.ini) パラメータ・リファレンス" を参照してください。Web ゲートウェイ・コンテナで提供される CSP.conf ファイルの内容の詳細は、"推奨オプション : NSD を使用しない Apache API モジュール" および "Nginx での NSD の使用" を参照してください。)

初期バージョンの CSP.ini ファイルを準備するにはいくつかの方法があります。例えば、既存または以前の Web ゲートウェイ導入環境のファイルを変更して、それを Web ゲートウェイ・コンテナに使用できます。あるいは、1 つのコンテナで独自のファイルまたは提供されている基本的な既定のバージョンで起動し、管理ページを使用して必要に応じて構成を変更した後、構成ファイルをコピーして追加のコンテナでそれを使用することもできます。"Web ゲートウェイ・ガイド" では、Web ゲートウェイ構成に関する包括的な情報を提供しています。

Web ゲートウェイ・コンテナは、オプションで永続的な %SYS 機能のバージョンで実行することができます。これは、Web ゲートウェイの構成が格納されているコンテナ内の webgateway と呼ばれる永続的なデータ・ディレクトリを作成します ("Web ゲートウェイ・コンテナの実行オプション" を参照)。このオプションを使用すると、CSP.log ログ・ファイルと同様に、CSP.ini ファイルおよび CSP.conf ファイル (既定またはユーザ指定) がそのディレクトリにコピーされます。これによって、既存の構成を失うことなくコンテナをアップグレードできます。

### 5.5.3 複数の Web ゲートウェイ・コンテナの同期された再構成

専用の Web ゲートウェイ・コンテナの場合、Web ゲートウェイの構成は比較的シンプルで、InterSystems IRIS インスタンス専用の単一のサーバ・アクセス・プロファイルや、そのインスタンスが処理するアプリケーション専用のアプリケーション・アクセス・プロファイルが含まれます。このため、専用 Web ゲートウェイの再構成も、その管理ページにアクセスして必要な変更を加えるだけで簡単に行うことができます。

ただし、(Web サーバ層のように) 同じクラスタの一部として複数の Web ゲートウェイ・コンテナを導入する場合、その構成を同時に更新する方法が必要になる可能性もあります (特に、サーバ・アクセス・プロファイルとアプリケーション・アクセス・プロファイルで、InterSystems IRIS ノードを追加する場合やアプリケーション・パスを変更する場合など)。CSP.ini マージ機能は、そのための便利な方法を提供します。コンテナ作成時に独自の CSP.ini ファイルを指定し、"Web ゲートウェイ・コンテナの実行オプション" の説明に従って、マウントされた外部ボリュームでステージングすると、コンテナのファイル・システム、または永続的なデータ・ディレクトリ (存在する場合) のいずれかに一度コピーされます。その後で、ステージングしたファイルが元の場所に残っていて、そのファイルに Web ゲートウェイ・コンテナからアクセスできる場合、ファイルの更新は、[SYSTEM]/RELOAD=1 設定と共に、各 Web ゲートウェイ・インスタンスのアクティブな CSP.ini ファイルに自動的にマージされます (どちらかの場所で)。これにより、構成が 1 分以内に Web ゲートウェイによって再読み込みされます。このため、複数の Web ゲートウェイ・コンテナを導入する際、すべてのコンテナに対して、同じ外部ボリュームで同じステージングされる CSP.ini ファイルをマウントする場合、ステージングされるファイルを更新することで、すべてを同一かつ同時に再構成できます。この方法は、専用の Web ゲートウェイ、つまりアプリケーション接続を複数の InterSystems IRIS コンテナに分散している単一の Web ゲートウェイの構成にも同じように適切に機能します。

このマージ更新アプローチを使用する場合、単一の Web ゲートウェイ・コンテナであるか、複数のコンテナであるかに関係なく、他の方法によって (例えば、Web ゲートウェイ管理ページを使用して) インスタンスの CSP.ini ファイルが変更されないようにする措置を講じることが大切です。ファイルに変更を加える場合は、次のいずれかを実行します。

- 更新するフィールド以外のすべてを削除するなどして、他の方法によって変更したステージングされる CSP.ini ファイルからフィールドを削除します。
- 他の方法による更新をステージングされるファイルに適用します。例えば、管理ページを使用してファイルを変更した後、変更したファイルをステージング場所にコピーし、元のステージングされるファイルを上書きできます。ステージングされるファイルを複数のコンテナで使用する場合、これによって、1 つのインスタンスで必要な変更を加え、マージ更新によってそれらを残りのインスタンスに適用できます。

注釈　ステージングされる CSP.ini ファイルは、変更されない限りマージされません。永続的なデータ・ディレクトリが使用されている場合、最後にマージされたファイルのコピーが、永続的な %SYS ディレクトリの /webgateway/CSP.ini.last_merge に保持されます。

独自の CSP.conf ファイルを外部でステージングすると、複数の Web ゲートウェイ・コンテナに存在する Web サーバを前述の CSP.ini で説明したのと同じ方法で最新の状態に保つことができます。ファイルが、ステージングした元の場所にあり、Web ゲートウェイ・コンテナからアクセス可能である限り、各 Web ゲートウェイ・コンテナでその変更が監視され、変更が検出されると、適切なディレクトリにファイルが再コピーされて、Web サーバは構成を再読み込みして再起動します。これは、ステージングされた CSP.ini で Web ゲートウェイのサーバ・アクセス・プロファイルを更新したことによって、Web サーバに構成されているアプリケーション・パスも更新する必要がある場合に特に役立ちます。

### 5.5.4 Web ゲートウェイ・コンテナのセキュリティ

Web ゲートウェイ・インスタンスの保護には、以下の 2 つの主要なタスクがあります。

- Web ゲートウェイのサーバ・アクセス・プロファイルに構成されている InterSystems IRIS インスタンスへの Web ゲートウェイ接続の保護。これには、少なくとも、ターゲット・インスタンスで既存のユーザ・アカウントを使用して InterSystems IRIS に対する認証を行う必要があります。ただし、TLS 暗号化の追加を強くお勧めします。このトピックの詳細は、"Web ゲートウェイ・ガイド" の "InterSystems IRIS への Web ゲートウェイ接続の保護" および "TLS ガイド" の "TLS を使用して InterSystems IRIS に接続するための Web ゲートウェイの構成" を参照してください。

  注釈　また、Web クライアントと Web サーバ間の接続を TLS で保護することも強くお勧めします。

- Web ゲートウェイの管理ページへのアクセスの保護。これには、Web ゲートウェイに対する認証のほか、管理ページにアクセスできるホストの制限が必要です。このトピックの詳細は、"Web ゲートウェイ・ガイド" の "Web ゲートウェイの既定パラメータの構成" にある "Web ゲートウェイ管理ページの概要" および "セキュリティ" を参照してください。

Web ゲートウェイ・コンテナ (および既定で、ローカルにインストールされている Web ゲートウェイ) の既定の CSP.ini では、事前定義のユーザ・アカウントである **CSPSystem** がこれら両方の目的で使用されます。ただし、いずれの目的にも任意の認証情報を使用できます。InterSystems IRIS インスタンスへの接続を保護する際に、各サーバ・アクセス・プロファイルに構成した認証情報がその特定のインスタンスで有効である必要があります。そうでなければ、いずれの認証情報セットに対しても制限がない状態になります。また、**CSPSystem** の代わりに、その特定の目的で作成したアカウントを使用すると、認証情報に関する情報をより厳格に制限できるようになります。

"複数の Web ゲートウェイ・コンテナの同期された再構成" で説明しているように、複数の InterSystems IRIS インスタンスとやり取りする複数の Web ゲートウェイが含まれるクラスタ化された構成を導入する場合、InterSystems IRIS へのすべての接続に 1 つの認証情報セットを使用し、Web ゲートウェイへのすべての管理アクセスに別の認証情報セットを使用すると便利です (ただし、異なるセットを使用することで認証情報の再利用が減り、セキュリティは強化されます)。これを実装するための、かなり便利で、(他の予防策が守られている場合) かなり安全な方法は以下のとおりです。

- 構成マージの `[Actions]` パラメータ ("構成マージを使用した InterSystems IRIS の自動構成" の "セキュリティ・オブジェクトの作成、変更、削除" を参照) を使用して InterSystems IRIS を導入し、Web ゲートウェイの認証専用のアカウントを作成して、パスワードは `PasswordHash` パラメータを使用して暗号化します。継続的な監視とマージの必要性に合わせて、マウントされた外部ボリュームでマージ・ファイルをステージングします。
- 以下を指定するカスタムの CSP.ini ファイルで Web ゲートウェイ・コンテナを導入します。
  - サーバ・アクセス・プロファイル内の、InterSystems IRIS の導入時に作成された Web ゲートウェイへのアクセスの認証情報
  - 管理ページへのアクセスのための、別の認証情報セット
  - 管理ページにアクセスできる限定された IP アドレスのセット

  継続的な監視とマージの必要性に合わせて、マウントされた外部ボリュームで CSP.ini ファイルをステージングします。

CPF マージ・ファイルおよび CSP.ini ファイルを継続的に監視およびマージすることで、後で CSP.ini マージ機能を使用して、Web ゲートウェイ・コンテナの同期再構成を行うことができます。これには、両方のサーバ・アクセス認証情報 (InterSystems IRIS コンテナと Web ゲートウェイ・コンテナの両方) および Web ゲートウェイ管理の認証情報の定期的なパスワード変更や、InterSystems IRIS インスタンスでの新規アカウントの定期的な作成、およびそれに応じた CSP.ini の更新といったセキュリティのベスト・プラクティスが含まれます。

重要

1 つ以上の Web ゲートウェイ・コンテナをカスタムの CSP.ini ファイルで導入する場合は、導入前に必ずファイル内の 3 つのパスワードを暗号化する必要があります。

- `[SYSTEM]/Password` ー Web ゲートウェイ管理アクセスのパスワード
- `[<server>]/Password` ー InterSystems IRIS 認証パスワード (各サーバ・アクセス・プロファイルに 1 つずつ)
- `[<server>]/SSLCC_Private_Key_Password` ー TLS 秘密鍵パスワード (TLS を使用している場合) (各サーバ・アクセス・プロファイルに 1 つずつ)

Web ゲートウェイ管理アクセスのパスワード `[SYSTEM]/Password` は、CSP.ini ファイルに入力する前に暗号化しておく必要があります。各サーバ・アクセス・プロファイルにある 2 つのパスワードについては、以下のように 2 つのオプションがあります。

1. 使用する CSP.ini で Web ゲートウェイ・コンテナを導入します。
2. `[<server>]/Password`、`[<server>]/SSLCC_Private_Key_Password` のいずれかまたは両方について、以下のいずれかを入力します。
   - プレーン・テキストのパスワード。Web ゲートウェイの起動時、または `[SYSTEM]/RELOAD=1` 設定を CSP.ini ファイルに追加することによって実行中に Web ゲートウェイが再読み込みされるときに、パスワードは自動的に暗号化されます (スクリプトおよびその他の自動化された方法を使用する場合は、CSPpwd ユーティリティを使用して、CSP.ini ファイル内のすべてのプレーン・テキストのパスワードを暗号化できます。ただし、Windows プラットフォームではパスワードの解読に制限があります)。
   - UNIX® および Linux システムでの、中括弧で囲んだコマンド (例えば、`Password={sh /tmp/PWretrieve.sh}`)。Web ゲートウェイの起動時または再読み込み時に、そのコマンドが実行され、結果はフィールドの値としてのみメモリに格納されます。これによって、永続ストレージに対してパスワードをプレーン・テキストでコミットすることなく、クラウド・プラットフォームやサードパーティのシークレット・マネージャなどのソースから、パスワードを取得できるようになります。

"認証とパスワード" で説明しているように、InterSystems IRIS インスタンスでは、導入の一環として、または導入後すぐに、**`CSPSystem`** などの事前定義のアカウントの既定のパスワードを変更する必要があります。**`CSPSystem`** または CSP.ini ファイルのその他の事前定義のアカウントのいずれかを 使用する場合は、パスワードを CSP.ini ファイルに正確に追加できるように、導入後の暗号化されたパスワードへの安全なアクセスを確保する必要があります。

注釈

コンテナ化された Web ゲートウェイと InterSystems IRIS インスタンス間の接続を TLS を使用して保護する ("Web ゲートウェイ・ガイド" の "InterSystems IRIS への Web ゲートウェイ接続の保護" および "TLS ガイド" の "TLS を使用して InterSystems IRIS に接続するための Web ゲートウェイの構成" を参照) か、SSL モードを使用して Apache Web サーバを保護する場合、証明書を提供するには、パスワードを使用しないサーバ・キーを生成し、そのキーと証明書の両方を Docker シークレット (該当する場合は Kubernetes シークレット) としてマウントするのがベスト・プラクティスです。認証情報を更新する必要がある場合は、このシークレットを更新するだけです。パスワードを使用するサーバ・キーを作成して、そのパスワードをキーおよび証明書とは別のシークレットとしてマウントすることもできます。この手法により、サーバ・キー・パスワードを手動で指定することで復元可能性が損なわれたり、自動で取得できるようにパスワードを記録することでセキュリティが損なわれたりすることがなくなります。

# 5.6 InterSystems IRIS コンテナの実行

このセクションでは、Docker、およびこのドキュメントで説明されている iris-main オプションを使用して InterSystems IRIS コンテナを起動する方法の例をいくつか示します。以下のトピックがあります。

- コマンド行の例
- スクリプトの例
- Docker Compose の例
- Web ゲートウェイ・コンテナの実行オプション

注釈　このセクションで示しているサンプルの docker run コマンドには、各例に関連するオプションのみが含まれており、実際に含まれることになるオプションが省略されています (例えば、"永続的な %SYS を使用した InterSystems IRIS コンテナの実行" のコマンド例)。

ヒュージ・ページを使用するには、IPC_LOCK カーネル機能が必要です。この機能がないと、InterSystems IRIS 向けに構成する際に、ヒュージ・ページを割り当てることができません。ほとんどのコンテナ・ランタイム・エンジンは、コンテナの作成時に具体的に要求されない限りコンテナにこの機能を付与しません。IPC_LOCK 機能をコンテナに追加するには、--cap-add IPC_LOCK オプションを docker create コマンドまたは docker run コマンドに組み入れます。これは、以下のスクリプト例で示されています。

このドキュメント内の例で使用されているイメージ・タグは、例えば以下の `2023.2.0.299.0` のように、期限切れであることがあります。イメージのダウンロードを試みる前に、現在のイメージ・タグについて "InterSystems Container Registry の使用" を参照してください。

## 5.6.1 InterSystems IRIS コンテナの実行 : docker run の例

以下に、iris-main のオプションを使用して InterSystems IRIS コンテナを起動するための docker run コマンドの例を示します。

- "InterSystems IRIS コンテナのライセンス・キー" で説明しているように、インスタンスが動作できるように、必要な InterSystems IRIS ライセンス・キーをコンテナ内に取り込む必要があります。下に示す例は、以下を実行します。
  - インスタンスのスーパーサーバのポート (1972) をホストのポート 9092 に公開します。
  - 永続的な %SYS の必要なオプションを含めます ("永続的な %SYS が指定され、マウントされていることの確認" を参照)。
  - iris-main -key オプションを使用します。ここで、ライセンス・キーは、永続的な %SYS ディレクトリ用にマウントされたボリューム上の **key/** ディレクトリ (つまり、コンテナ内の外部ストレージ **/dur/key/** の **/data/durable/key/**) に置かれ、InterSystems IRIS インスタンスの起動前に、永続的な %SYS ディレクトリ内の **mgr/** ディレクトリ (コンテナ内の外部ストレージ **/dur/iconfig/mgr/** の **/data/durable/iconfig/mgr/**) にコピーされます。ライセンス・キーは **mgr/** ディレクトリ内にあるため、インスタンスの起動時に自動的に有効になります。

```
docker run --name iris11 --detach --publish 9092:1972
  --volume /data/durable:/dur
  --env ISC_DATA_DIRECTORY=/dur/iris_conf.d
  intersystems/iris:latest-em --key /dur/key/iris.key
```

- この例では、永続的なデータ・ボリュームでステージングされ、InterSystems IRIS インスタンスの最初の起動前にそのインスタンスの CPF にマージされる設定が含まれる構成マージ・ファイルを追加しています ("InterSystems IRIS コンテナの自動導入" を参照)。これは例えば、"コンテナ化された InterSystems IRIS のファイル・システムの分離" で説明しているように、既定では永続的な %SYS ツリー内で同一のディレクトリであるインスタンスのプライマリおよび

代替ジャーナル・ディレクトリ (CPF 内の [Journal]/CurrentDirectory および AlternateDirectory) が別個のファイル・システム上に配置されるよう再構成するために使用できます。

```
docker run --name iris17 --detach --publish 9092:1972
  --volume /data/durable:/dur
  --env ISC_DATA_DIRECTORY=/dur/iris_conf.d
  --env ISC_CPF_MERGE_FILE=/dur/merge/merge.cpf intersystems/iris:latest-em
  --key /dur/key/iris.key
```

ステージング・ディレクトリは、この場合はどちらも永続的な %SYS に対してマウントされたボリュームに配置されており、これらのディレクトリが同じであるか、同じライセンスを含む必要があります。

注釈　"InterSystems IRIS イメージのダウンロード" で説明した InterSystems IRIS Community Edition イメージには無料の一時ライセンスが付属しているので、--key オプションはこのイメージには使用できません。

### 5.6.2 InterSystems IRIS コンテナの実行：スクリプトの例

以下のスクリプトは、テストを目的として InterSystems IRIS コンテナをすばやく作成して起動するために記述されています。このスクリプトには、"InterSystems IRIS コンテナのライセンス・キー" で説明しているように、ライセンス・キー内でコピーを行うための iris-main --key オプションが組み込まれています。

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

# Definitions to toggle_____________________________________________
container_image="intersystems/iris:latest-em"

# the docker run command
docker run -d
  --name iris
  --hostname iris
# expose superserver port
  -p 9091:1972
# mount durable %SYS volume
  -v /data/durable:/dur
# specify durable %SYS directory and CPF merge file
  --env ISC_DATA_DIRECTORY=/dur/iris_conf.d
  --env ISC_CPF_MERGE_FILE=/dur/merge/merge.cpf
# enable allocation of huge pages
  --cap-add IPC_LOCK
  $container_image
  --key /dur/key/iris.key
```

### 5.6.3 InterSystems IRIS コンテナの実行：Docker Compose の例

Docker Compose は、複数コンテナの Docker アプリケーションを定義および実行するツールであり、Docker とのコマンド行でのやり取りに代わる方法を提供します。Compose を使用するには、作成、起動、および管理するコンテナの仕様が含まれる docker-compose.yml を作成してから、docker-compose コマンドを使用します。詳細は、Docker ドキュメンテーションの "Overview of Docker Compose" をまずは参照してください。

以下は compose.yml ファイルの例です。前述のスクリプトと同様に、このファイルにはこのドキュメントで説明している要素のみが含まれています。

```
version: '3.2'

services:
  iris:
    image: intersystems/iris:latest-em
    command:  --key /dur/key/iris.key
    hostname: iris
    ports:
    # the superserver port
    - "9091:1972"
    volumes:
    # the durable %SYS volume
    - /data/durable:/dur
    environment:
    # the durable %SYS directory
    - ISC_DATA_DIRECTORY=/dur/iris_conf.d
    # the CPF merge file
    - ISC_CPF_MERGE_FILE=/dur/merge/merge.cpf
```

### 5.6.4 Web ゲートウェイ・コンテナの実行オプション

プロダクション内で、複数の Web ゲートウェイ・コンテナを含む複数コンテナの InterSystems IRIS クラスタを導入するには、InterSystems Kubernetes Operator (IKO) を使用する方法をお勧めします ("InterSystems IRIS コンテナの自動導入" を参照)。バージョン 3.6 の IKO では、管理ポータルへのアクセス専用の Web ゲートウェイ・コンテナの導入は、各 InterSystems IRIS ポッドにサイドカー・コンテナとして追加することで実行されます。詳細は、"InterSystems Kubernetes Operator の使用" を参照してください。

開発とテストを目的として、管理ポータルへのアクセス専用の Web ゲートウェイ・コンテナを使用した InterSystems IRIS コンテナの導入には、以下のような 3 つの基本的な方法があります。

- Docker Compose およびその他のスクリプト作成ツール
- Kubernetes
- InterSystems IRIS インスタンスの他に、Web ゲートウェイ・インスタンスと Web サーバを含むユーザ作成の InterSystems IRIS イメージ

これらの各方法の例は、https://github.com/intersystems-community/webgateway-examples/tree/master を参照してください。必要なすべてのファイルといくつかの有用なスクリプトも参照できます。ユーザが作成する InterSystems IRIS イメージの一般的な説明は、"InterSystems IRIS イメージの作成" を参照してください。このセクションの残りの部分では、Web ゲートウェイ・コンテナの実行時に使用可能なコマンド行オプションについて説明します。

Web ゲートウェイ・コンテナの作成および起動時に唯一必要なオプションは、以下のコマンドに示すとおり、ホスト・ポートへのコンテナ・ポート 80 と 443 の公開です。これによって、その他のエンティティが Web ゲートウェイと Web サーバにアクセスできます。

```
docker run -d --publish 80:80 --publish 443:443
  containers.intersystems.com/intersystems/webgateway:latest-em
```

以下はオプションです。

- 永続的なデータ・ディレクトリ ー Web ゲートウェイの構成ファイルが格納されているコンテナ内に webgateway と呼ばれる永続的なデータ・ディレクトリを作成するには、--volume オプションを使用して永続データ・ボリュームをマウントし、**ISC_DATA_DIRECTORY** 環境変数を使用してコンテナ内でのそのディレクトリの場所を指定します ("Web ゲートウェイの構成" を参照)。例えば、次のコマンドは、コンテナ内の /dur/webgateway と、ホストのファイル・システム上の /nethome/user21/dur/webgateway に永続的なデータ・ディレクトリを作成します。

  ```
  docker run -d --name wg11 --publish 80:80 --publish 443:443
    --volume /nethome/user21/dur:/dur --env ISC_DATA_DIRECTORY=/dur
     containers.intersystems.com/intersystems/webgateway:latest-em
  ```

これは、InterSystems IRIS コンテナの永続的な %SYS を有効にする手順と同等です。これらのオプションと永続的な %SYS の全般的な使用法は、"永続インスタンス・データを保存するための永続的な %SYS" を参照してください。

重要　**webgateway-lockeddown** イメージには、ユーザ **irisowner** (UID 51773) によってインストールおよび所有されている Web ゲートウェイと Web サーバが含まれているため、ロック・ダウン・コンテナの永続的なデータ・ディレクトリに指定するホストのファイル・システムの場所は、**irisowner** によって書き込み可能である必要があります (これを有効にするには、多くの場合、root 特権が必要です)。

永続的なデータのオプションで **webgateway** コンテナを実行すると、以下の状況が生じます。

- 指定された外部ボリュームがマウントされます。
- **ISC_DATA_DIRECTORY** で指定された場所に webgateway ディレクトリが存在しない場合、Web ゲートウェイで使用できるように、次のようにこれが作成され、そこに構成ファイルがコピーされます。
  - 提供した構成ファイルは、コンテナ内の既定の場所へのリンクと共に、webgateway ディレクトリにコピーされます。
  - 提供していない構成ファイルは、コンテナ内の既定の場所から、それらの場所へのリンクと共にコピーされます。
- webgateway ディレクトリが既に **ISC_DATA_DIRECTORY** で指定された場所に存在する場合、これは、前の **webgateway** コンテナのデータ・ディレクトリと見なされます。
  - これに期待される Web ゲートウェイ構成ファイルが含まれる場合、これらはコンテナ内のそれらの場所にリンクされ、Web ゲートウェイにより使用されます。ユーザが提供する構成ファイルを指定するオプション (以下で説明) は無視されます。

    重要　既存の永続的な webgateway ディレクトリを使用して **webgateway** コンテナを導入する際は、独自の構成ファイル (以下で説明) を提供できないため、同じ操作でコンテナ化された Web ゲートウェイをアップグレードおよび再構成することはできません。代わりに、前のコンテナの永続的な webgateway ディレクトリを使用して新しいバージョンのコンテナを導入することから開始し、必要に応じて Web ゲートウェイを再構成します。
  - 期待される 1 つ以上の Web ゲートウェイ構成ファイルが存在しない場合は、webgateway が破損していると見なされます。エラーがログに記録され、コンテナは起動に失敗します。

    重要　**webgateway-lockeddown** イメージを使用して、永続データ・ディレクトリを使用する **webgateway** コンテナをバージョン 2021.2 以降にアップグレードする場合、既存の永続ディレクトリを、ユーザ **irisowner** によって書き込み可能にする必要があります。

コンテナ内の Web ゲートウェイにアクセスできる書き込み可能な永続ストレージ上の場所を指定するのに **ISC_DATA_DIRECTORY** 変数を使用しない場合、既定で、永続的なデータ・ディレクトリは作成されず、構成ファイルは既定の場所に維持されます (前述のとおり)。

- ユーザ定義の構成ファイル – 独自の CSP.ini ファイルを提供するには、--volume オプションを使用して永続データ・ボリューム (**ISC_DATA_DIRECTORY** によって指定された永続的なデータ・ディレクトリ・ボリュームが使用されている場合、それとは別個の) をマウントしてそのボリューム上でファイルをステージングし、**ISC_CSP_INI_FILE** 環境変数を使用してその場所を示します。前述のとおり、永続的な webgateway ディレクトリも作成する場合は、このファイルがそのディレクトリにコピーされ、コンテナ内の元の場所にリンクされます。このディレクトリを作成しない場合は、このファイルは元の場所にコピーされ、既定のファイルが上書きされます。

  重要　マージ更新アプローチを使用する場合 ("Web ゲートウェイの構成" を参照)、ステージングされる CSP.ini ファイルが元の場所に維持され、コンテナからアクセスできる必要があります。

独自の CSP.conf ファイルを提供するには、同じことを実行し、**ISC_CSP_CONF_FILE** 環境変数を使用してその場所を指定します。例えば、永続的な webgateway ディレクトリを作成し、独自の CSP.ini および CSP.conf ファイルを Web ゲートウェイの構成に使用するよう指定するには、次のようなコマンドを使用します。

```
docker run -d --name wg11 --publish 80:80 --publish 443:443
  --volume /nethome/user21/dur:/dur --env ISC_DATA_DIRECTORY=/dur
  --volume /nethome/user21/config:/config --env ISC_CSP_INI_FILE=/config/CSP.ini
  --env ISC_CSP_CONF_FILE=/config/CSP.conf
  containers.intersystems.com/intersystems/webgateway:latest-em
```

これらの変数のいずれかが指定されていない場合は、既定で、コンテナ内にある基本の既定ファイルを使用します。

- Apache Web サーバの SSL モジュールを有効にするには、(イメージの仕様に従って) --ssl エントリポイント・オプションを追加します。オプションが省略された場合の既定では、Apache SSL モジュールは有効化されません。
- Web ゲートウェイ・サーバを Apache に特定するには、--server server-name エントリポイント・オプションを使用します。オプションが省略された場合の既定では、コンテナの ID を使用します (Docker の --hostname オプションが含まれていない場合。含まれている場合は指定された値が使用されます)。

以下は、説明したすべてのオプションを使用した docker run コマンドの使用例です。

```
docker run -d --name wg11 --publish 80:80 --publish 443:443
  --volume /nethome/user21/dur:/dur --env ISC_DATA_DIRECTORY=/dur
  --env ISC_CSP_INI_FILE=/dur/CSP.ini --env ISC_CSP_CONF_FILE=/dur/CSP.conf
  containers.intersystems.com/intersystems/webgateway:latest-em
  --ssl --server webgateway11
```

## 5.7 InterSystems IRIS コンテナのアップグレード

コンテナ化されたアプリケーションをアップグレードしたり、他の方法で変更したりする場合には、既存のコンテナを削除するか名前変更し、docker run コマンドで別のイメージをインスタンス化することによって、新規コンテナを作成して起動します。目的はアプリケーションを変更することであり、従来のアプリケーションならアップグレード・スクリプトを実行したりプラグインを追加したりして変更を行いますが、この場合の新規アプリケーション・インスタンスは前のものと実際には本来の関連がありません。代わりに、同じアプリケーションの各バージョンを表す、別々のイメージから作成される別々のコンテナ間の連続性を維持するものは、コンテナ外部の環境との間で確立される相互作用です。例えば、docker run コマンドの --publish オプションによってホストに対して公開するコンテナ・ポート、--network オプションによってコンテナを接続する先のネットワーク、アプリケーション・データを永続化するために --volume オプションによってコンテナ内でマウントする外部ストレージ場所などがこれに当たります。

重要　永続的な %SYS を使用して InterSystems IRIS コンテナをバージョン 2021.2 以降にアップグレードする前に、既存の永続ディレクトリを、例えばこのコマンドで、ユーザ 51773 によって書き込み可能にする必要があります。

```
chown -R 51773:51773 root-of-durable-%SYS-directory
```

これを変更するには、多くの場合、root 特権が必要です。

InterSystems IRIS では、インスタンス固有データを永続化するための永続的な %SYS 機能によってアップグレードを可能にしています。アップグレードされたコンテナ内のインスタンスは、元のインスタンスの永続的な %SYS のストレージ位置を使用し、かつ同じネットワークの場所を使用する限り、実質的に元のインスタンスを置き換えて、InterSystems IRIS をアップグレードします。インスタンス固有データのバージョンが新規インスタンスのバージョンに一致しない場合、永続的な %SYS は必要に応じてデータをインスタンスのバージョンにアップグレードします (永続的な %SYS の詳細は、"永続インスタンス・データを保存するための永続的な %SYS" を参照してください)。

新規コンテナを起動する前に、元のコンテナを削除するか、停止して名前変更する必要があります。

注意　ベスト・プラクティスとして元のコンテナを削除することをお勧めします。アップグレードの後に元のコンテナが起動すると、InterSystems IRIS の 2 つのインスタンスが同一の永続的な %SYS データを使用しようとして、データ損失など予測不能な動作が生じる可能性があるためです。

通常、アップグレード・コマンドは元のコンテナの実行に使用するコマンドと同じですが、イメージ・タグが異なります。以下の docker run コマンドでは、元のコンテナを作成した docker run コマンドと、アップグレードされたコンテナを作成するコマンドで異なるのは、version_number の部分のみです。

```
$ docker stop iris
$ docker rm iris
$ docker run --name iris --detach --publish 9091:1972
  --volume /data/durable:/dur --env ISC_DATA_DIRECTORY=/dur/iconfig
  intersystems/iris:<version_number> --key /dur/key/iris.key
```

アップグレードするインスタンスが正常にシャットダウンされていないことを永続的な %SYS で検出した場合、アップグレードは続行できません。このようなインスタンスを起動する場合、データ整合性を確保するために手動で WIJ とジャーナルのリカバリを実行する必要があるためです。これを修正するには、″データ整合性ガイド″ の “バックアップとリストア” の章の ″自動 WIJ およびジャーナル・リカバリを使用しない InterSystems IRIS の起動″ で説明されている手順を使用して、インスタンスを起動してから正常にシャットダウンする必要があります。コンテナが動作中の場合、これを行うには、コマンド docker exec -it container_name bash を実行してコンテナ内部でシェルを開き、説明されている手順に従います。ただし、コンテナが停止している場合は、データ整合性が損なわれるおそれがあるため、インスタンスを自動的に再起動せずにコンテナを起動することはできず、シェルを開くことができません。この場合、以下の手順を使用して、コンテナを再起動する前に正常なシャットダウンを行ってください。

1. 元のコンテナの作成に使用したものと同じコマンドを使用して複製コンテナを作成します。これには、同じ永続的な %SYS の場所と同じイメージを指定することを含みます。ただし、iris-main -up false オプションを追加します (″iris-main プログラム″ を参照してください)。このオプションにより、コンテナの起動時にインスタンスが自動的に起動しないようにします。
2. コマンド docker exec -it container_name bash を実行して、コンテナ内部でシェルを開きます。
3. ″自動 WIJ およびジャーナル・リカバリを使用しない InterSystems IRIS の起動″ で説明されている手順に従います。
4. リカバリと起動が完了したら、iris stop instance_name. (インターシステムズ提供のコンテナ内のインスタンスは、常に **IRIS** と名付けられます) を使用して、インスタンスをシャットダウンします。
5. 元のコンテナを起動します。このコンテナは、複製コンテナ内で安全に回復された永続的な %SYS データを使用しているため、通常の方法で起動しても安全です。

注釈　永続的なデータ・ディレクトリで Web ゲートウェイ・コンテナをアップグレードする方法の詳細は、″Web ゲートウェイ・コンテナの実行オプション″ を参照してください。

## 5.8 InterSystems IRIS イメージの作成

ほとんどのユース・ケースにおいて、InterSystems IRIS のカスタムの Docker イメージを作成する、最も単純でエラーが発生しにくいアプローチは、InterSystems IRIS が既にエントリポイントとして iris-main プログラムを使用してインストールされている状態で、Dockerfile のベースをインターシステムズが提供する既存のイメージとするというものです。その後、関連する依存関係を独自のアプリケーション・ソリューションに追加し、InterSystems IRIS インスタンスを起動します。オプションで構成マージ機能を使用し、その構成を変更して、これに対して他のコマンドを発行します。

例えば、アプリケーション、およびアプリケーションが必要とする事前定義の 2 つのデータベースを含む、InterSystems IRIS を作成するとします。Dockerfile を作成すると、以下を実行できます (手順の後の例に示されているとおり)。

1. ベースとして InterSystems IRIS イメージを開始します。
2. ユーザ **root** に切り替えて、ベースのサードパーティの依存関係をアップグレードします。

重要　ベースでのパッケージのアップグレードが、セキュリティの脆弱性を回避するためのベスト・プラクティスです。

3. インスタンス所有者ユーザ **irisuser**/**51773** に切り替えます (″所有権とディレクトリ″ を参照)。
4. アプリケーション・コードを含むファイルをコピーします。
5. 構成マージ・ファイルをコピーし、インスタンスで必要なネームスペースとアプリケーション・データベースを作成します。また、″認証とパスワード″ で説明したとおり、PasswordHash パラメータを使用して既定のパスワードを変更し、その他必要な構成変更を行います。マージ・ファイルは次のようになります。

```
[Startup]
PasswordHash=dd0874dc346d23679ed1b49dd9f48baae82b9062,10000,SHA512
[Actions]
CreateDatabase:Name=DB-A,Directory=/usr/irissys/mgr/DB=A,Size=5368,MaxSize=536871,ResourceName=%DB_%DB-A
CreateDatabase:Name=DB-B,Directory=/usr/irissys/mgr/DB=B,Size=5368,MaxSize=536871,ResourceName=%DB_%DB-B
CreateNamespace:Name=DB-A,Globals=DB-A,Routines=SYS
CreateNamespace:Name=DB-B,Globals=DB-B,Routines=SYS
```

6. InterSystems IRIS インスタンスを開始します。
7. 構成マージ・ファイルを使用して iris merge コマンドを実行し、データベースの作成を含め、インスタンスを再構成します。
8. インスタンスをシャットダウンします。
9. 準備した IRIS.DAT データベース・ファイルをコピーし、前の手順で作成したこれらのファイルを上書きします。
10. 今後 **ISC_CPF_MERGE_FILE** を使用してインスタンスのマージ・ファイルとして指定する予定がある場合を除き、マージ・ファイルを削除します。

以下の Dockerfile のサンプルは、上記の手順を示しています。

```
FROM intersystems/iris:latest-em

USER root

RUN apt-get update && apt-get -y upgrade \
 && apt-get -y install unattended-upgrades

USER 51773

COPY application.xml /

COPY merge.cpf /tmp/

RUN iris start IRIS \
 && iris merge IRIS /tmp/merge.cpf \
 && iris stop IRIS quietly

COPY DB-A.DAT /usr/irissys/mgr/DB-A
COPY DB-B.DAT /usr/irissys/mgr/DB-B

RUN rm /tmp/merge.cpf
```

重要　このセクションで説明しているように独自の InterSystems IRIS イメージのビルドを試行する前に、3 つのセクションで提供される情報に精通するようにしてください。それらのセクションでは、InterSystems IRIS で提供される iris-main プログラム、永続的な %SYS 機能、およびコンテナ化ツールについて説明しています。

注釈　Docker イメージを作成する場合の重要な考慮事項はイメージ・サイズです。イメージが大きいほど、ダウンロードにかかる時間が長くなり、ターゲット・マシンにより多くのストレージが必要になります。イメージ・サイズの管理の良い例として、InterSystems IRIS ジャーナル・ファイルおよびライト・イメージ・ジャーナル (WIJ) があります。"永続インスタンス・データを保存するための永続的な %SYS" で説明しているように、これらのファイルがコンテナ外部の永続ストレージに再配置されていることを前提として (これらのファイルはここに再配置する必要があります)、これらのファイルをコンテナ内のインストール済みの InterSystems IRIS インスタンスから削除することによって、InterSystems IRIS またはアプリケーション・イメージのサイズを削減できます。

例に含まれていたようなユーザ定義のデータベースを、永続的な %SYS で作成された永続的なデータ位置に複製するには、コピーしたデータベース・ファイルが書き込み可能である必要があります。

何らかの理由で InterSystems IRIS のインストール後に messages.log ファイルまたは console.log ファイルの内容を削除して、コンテナ内でインスタンスが起動したときにこれらのファイルのどちらかまたは両方を空にしたい場合、これらのファイルを削除しないでください。iris-main プログラムでは、存在しないログ・ファイルを致命的なエラーとして扱うためです。その代わりに、これらのファイルを空にして、サイズをゼロにすることができます。

このアプローチの詳細を確認する際には、"InterSystems IRIS Docker イメージのダウンロード" で説明している InterSystems IRIS Community Edition イメージを基本イメージとして使用できます。ただし、これには機能制限があることに留意してください。

このアプローチを使用して、Web ゲートウェイ・インスタンスと Web サーバを含む InterSystems IRIS イメージを作成する方法の例は、https://github.com/intersystems-community/webgateway-examples/tree/master/demo-dockerfile を参照してください。

## 5.9 InterSystems IRIS のコンテナ化ツール

インターシステムズは、InterSystems IRIS ベースのユーザ独自のコンテナ・イメージの作成をサポートするために、いくつかのコンテナ化ツールを提供しています。このセクションでは、以下のトピックについて説明します。

- 必須の環境変数
- SYS.Container API およびイメージ・ビルド・スクリプト

### 5.9.1 必須の環境変数

UNIX および Linux への InterSystems IRIS インスタンスの自動インストールを構成する際に使用できる多数のインストール・パラメータがあります。これらの使用法は、"インストール・ガイド" の "InterSystems IRIS の自動インストール" を参照してください。"InterSystems IRIS イメージの作成" で説明しているように、インターシステムズ提供のイメージを基本として使用するのではなく、Dockerfile でキットから InterSystems IRIS インスタンスをインストールする場合、コンテナの実行時環境で環境変数として必要なインストール・パラメータもイメージに組み込む必要があります。これらの変数がないと、イメージからのコンテナの作成は失敗します。これらの変数は、インターシステムズ提供のすべてのイメージに含まれています。これらの変数を、インターシステムズによって設定されている値と共に、以下のテーブルに示します。

テーブル 1: 環境変数としてコンテナ化に必要なインストール・パラメータ

| パラメータ/変数 | 説明 | インターシステムズの値 |
|---|---|---|
| ISC_PACKAGE_INSTANCENAME | インストール対象のインスタンスの名前 | IRIS |
| ISC_PACKAGE_INSTALLDIR | インスタンスのインストール先のディレクトリ | /usr/irissys |
| ISC_PACKAGE_IRISUSER | InterSystems IRIS スーパーサーバの実効ユーザ | irisowner |
| ISC_PACKAGE_IRISGROUP | InterSystems IRIS プロセスの実効ユーザ | irisowner |
| ISC_PACKAGE_MGRUSER | インストール所有者のユーザ名 | irisowner |
| ISC_PACKAGE_MGRGROUP | インスタンスの開始および停止の権限を持つグループ | irisowner |

注釈　独自の InterSystems IRIS イメージを構築する場合、オプションで IRISSYS 変数を設定して、レジストリ・ディレクトリを指定できます。インターシステムズではすべてのイメージでこれを /home/irisowner/irissys に設定します。この環境変数が含まれていない場合、レジストリ・ディレクトリは /usr/local/etc/irissys です。

ここに示した環境変数は、"所有権とディレクトリ" で説明している構成の詳細の指定に使用されます。

### 5.9.2 SYS.Container API

InterSystems IRIS イメージをビルドする際に、インターシステムズでは SYS.Container API を使用して、インストールした InterSystems IRIS インスタンスをコンテナ・イメージに安全にシリアル化できる状態にします。このクラスには、個々に使用できるメソッドがいくつか含まれていますが、それらのうち、SYS.Container.QuiesceForBundling() は、1 回の処理で必要なメソッドすべてを呼び出すため、インターシステムズではイメージの作成にこれを使用しています。このアプローチを使用することをベスト・プラクティスとしてお勧めします。Linux シェル/ObjectScript 境界を越えてのエラーチェックは難しいうえ、InterSystems IRIS から通知なしのエラーが生じるリスクがあるためです。呼び出しが少ないほど、このリスクも小さくなります。

SYS.Container コードは、Linux プラットフォームにインストールされたあらゆる InterSystems IRIS インスタンスに含まれており、完全に表示可能です。ドキュメントについては、"クラス・リファレンス" を参照してください。メソッドには以下のようなものがあります。

- SYS.Container.QuiesceForBundling()

  InterSystems IRIS をコンテナ・イメージに安全にシリアル化できる状態にするために必要なすべての ObjectScript コードを呼び出します。

- SYS.Container.ChangePassword()

  1 つ以上のロールが割り当てられている有効なユーザ・アカウントすべてに対するパスワードを変更します。"認証とパスワード" で説明されているように、iris-main --password-file オプションとパスワード変更スクリプトによって呼び出されます。

- SYS.Container.ChangeGatewayMgrPassword()

  Web ゲートウェイ管理パスワードを変更します ("Web ゲートウェイ・ガイド" の "Web ゲートウェイ管理ページの概要" を参照)。"認証とパスワード" で説明されているように、iris-main --password-file オプションとパスワード変更スクリプトによって呼び出されます。

- SYS.Container.ForcePasswordChange()

  1 つ以上のロールが割り当てられている有効なユーザ・アカウントで **[次回ログイン時にパスワード変更]** を設定します (″ユーザ・アカウントのプロパティ″ を参照)。

- SYS.Container.KillPassword()

  指定したユーザに対してパスワードベースのログインを無効にします。その他の形式の認証 (″認証：身元の確認″ を参照) は有効のままになります。

- SYS.Container.EnableOSAuthentication()

  インスタンスに対し OS ベースの認証を有効にします (″オペレーティング・システム・ベースの認証について″ を参照してください)。

- SYS.Container.SetNeverExpires()

  指定したユーザ・アカウントに **[アカウントを有効期限切れにしない]** を設定します。これを設定しないと、イメージに設定してから 90 日以上経過したユーザ・アカウントは期限切れになります (″ユーザのプロパティ″ を参照してください)。

- SYS.Container.PreventFailoverMessage()

  新たに起動したコンテナ内のインスタンスからのジャーナル・ロールオーバー・メッセージを阻止します。

- SYS.Container.PreventJournalRolloverMessage()

  インスタンスが実行されているホストの名前が前回の実行時に格納されたホスト名と同じではないためにインスタンスが警告を発しないようにします。

- SYS.Container.SetMonitorStateOK()

  システム・モニタからのレベル 1 およびレベル 2 のアラートを削除し、レベル 3 のアラートがある場合はエラーを生成します (″監視ガイド″ の “システム・モニタの使用” の章を参照してください)。

注釈　ここにリストしたメソッドは、″認証とパスワード″ で説明している構成の詳細の指定に使用できます。

一般的なのは、インスタンスの起動時に、iris terminal コマンドを介してこれらのメソッドを呼び出すことで、インターシステムズが提供する InterSystems IRIS イメージ (″InterSystems IRIS イメージの作成″ を参照) に基づく Dockerfile にこれらを含めるアプローチです。

```
RUN iris start $ISC_PACKAGE_INSTANCENAME \
    && iris terminal $ISC_PACKAGE_INSTANCENAME -U %SYS
"##class(SYS.Container).PreventJournalRolloverMessage()"
    && iris terminal $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Container).SetMonitorStateOK()"
    && iris terminal $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Container).QuiesceForBundling()"
    && iris terminal $ISC_PACKAGE_INSTANCENAME quietly
```

# 6 Docker/InterSystems IRIS に関するその他の考慮事項

このセクションでは、InterSystems IRIS イメージのコンテナ・イメージを作成および実行する際に留意する、以下のようなその他の考慮事項について説明します。

- 分離したパーティションへのイメージ・ストレージの配置
- Docker ブリッジ・ネットワーク IP アドレス範囲の競合

## 6.1 分離したパーティションへのイメージ・ストレージの配置

Docker コンテナ・イメージの既定のストレージ場所は /var/lib/docker です。これはルート・ファイル・システムの一部なので、分離したパーティションにマウントすると役に立つ場合があります。この目的は、急激なストレージ不足を防ぐことと、ファイル・システムを破損から保護することの両方です。これらの問題が生じると、Docker と OS の両方のリカバリが困難になることがあります。例えば、SUSE の記述によると、ファイル・システムの破損が生じた場合、Docker ホストのオペレーティング・システムに影響が及ばないように、/var/lib/docker は別のパーティションまたはボリュームにマウントすることをお勧めしています。

良い方法の 1 つは、Docker エンジンのストレージ設定をこの代替ボリューム・パーティションに設定することです。例えば、Fedora ベースのディストリビューションでは、Docker デーモン構成ファイルを編集して (Docker ドキュメンテーションの "Configure and troubleshoot the Docker daemon" を参照)、Docker エンジンの ExecStart= コマンド行オプションを見つけ、引数として - を追加します。

## 6.2 Docker ブリッジ・ネットワーク IP アドレス範囲の競合

コンテナ・ネットワークの場合、Docker では、既定で、サブネット 172.17.0.0/16 でブリッジ・ネットワークが使用されます (Docker ドキュメントの "Use bridge networks" を参照)。このサブネットを既にネットワークで使用している場合は、競合が発生して、Docker が起動しなくなったり、ネットワーク・エラーが発生したりすることがあります。

これを解決するには、使用している IP アドレスと競合しない範囲にサブネットを割り当て直すように、Docker 構成ファイルのブリッジ・ネットワークの IP 構成を編集できます (ユーザの組織の IT 部門がこの値の指定を支援できる場合があります)。この変更を行うには、Docker デーモン構成ファイル (既定では /etc/docker/daemon.json) に以下のような行を追加します。

```
"bip": "192.168.0.1/24"
```

daemon.json ファイルのコンテンツに関する詳細は、Docker ドキュメントの "Daemon configuration file" に記載されています。また、"Configure and troubleshoot the Docker daemon" も参照してください。