Migration Guide

Payara Server 4 から 5 への移行ガイド

The Payara® Platform - Production-Ready,
Cloud Native and Aggressively Compatible.

Contents

はじめに

Payara Server 5 は 2018 年の初めに Payara Server の次期メジャー・バージョンとして登場しました。Payara Server 4 は GlassFish Open Source Edition 4.1 から派生したもので、既存の GlassFish サーバーをそのまま置き換えようとしている数多くのユーザーにお勧めできる選択肢でした。Payara Server はお客様とコミュニティからいただいた製品の使い勝手に関するフィードバックをもとに、その期待に応えるべく革新的なものへと進化してきました。Payara Server 4 は市場において、開発者および運用担当者の双方にとって信頼ある選択肢ではありますが、現在の環境で要求されている商用水準での利用に向けた改良と変更を加える余地がある、と私達は結論を下しました。そのため、Payara Server 5 では、GlassFish Server Open Source Edition 5 から派生しつつも、ユーザーの生産性や機能性へのニーズに合わせて、一部の機能および内部メカニズムを GlassFish の実装とは異なるものにしています。そのために、Payara Server 5 では以下の目標を設定しています。

- クラウドおよびコンテナとの高い親和性
- 最新の Java API への準拠
- 古いコンポーネントやサードパーティ・ライブラリへの依存の軽減
- デプロイされたアプリケーションの全般的なパフォーマンスおよび品質の向上
- 最高水準のセキュリティと監視機能の提供

このガイドは、Payara Server 4 から Payara Server 5 へ移行する際に直面すると考えられる主な課題を解決するための準備や理解の手助けとなることを目的としています。プロジェクトの移行計画を立てる前に、個々のケースで行うことになる作業手順をより深く理解するため、このドキュメント全体を熟読していただきたく思います。Payara Enterprise または Migration & Project Support 契約をお持ちの商用サポートのお客様は、移行に関する疑問点を解決するためサポート・チケットをご利用になれます。さらに移行プロセス全体のハンズオン・ガイダンスをご希望の場合には、Payara Accelerator (コンサルティング・サービス) の一環としてアップグレード・サービスをご検討いただくことも可能です。Payara Server 4 を商用サポート契約なしでご利用されている場合には、アプリケーション・サーバーの移行プロセスを円滑に進められるよう、固定価格の Migration & Project Support を含むサポート・オプションもご検討いただけますと幸いです。

Payara Server 5 の主な特長

進化したクラスタおよび高可用性

高可用性は多くの開発者とサーバー管理者にとって馴染み深い概念です。ミッション・クリティカルおよび高パフォーマンスを要求されるアプリケーションやサービスにとって、ビジネスがエラーや高負荷の影響を受けないために高可用性戦略を採ることは必須事項となります。Payara Server は以下を実現するためにドメイン・データ・グリッドの概念を備えています。

- ドメインのすべてのインスタンス間でデータを共有し、フェール・オーバーのためデータを複製する
- ドメインのすべてのインスタンスの構成を集中管理する

ドメイン・データ・グリッド上では、アプリケーションとリソースはデプロイメント・グループと呼ばれる複数のグループに含めることができます。これには以下の機能が提供されます。

- デプロイ「対象」としての機能、すなわちデプロイメント・グループにデプロイされるアプリケーションとリソースは、グループ内のすべてのインスタンスに自動的にデプロイされる
- ドメイン内の複数のインスタンスを単一の操作で制御することが可能 (例: グループのすべてのインスタンスの開始/停止)

これは高可用性サービスを持つ Hazelcast との優れた統合と、Hazelcast ベース・クラスタの構成しやすさによるものです。さらに柔軟なクラスタリング・オプションも提供されます。オプションの組み合わせが簡単になったことで、スケーラブルな環境に適したクラスタを動的に形成し、インスタンスとアプリケーションのデプロイをデプロイメント・グループによってコントロールできるようになっています。

ドメイン・データ・グリッドは Payara Server 4 の Hazelcast クラスタと比較することが可能で、デプロイメント・グループは Payara Server 4 のクラスタと似ています。しかし、デプロイメント・グループはドメイン・データ・グリッド上に構築され、Payara Server 4 のクラスタとは異なり Hazelcast によって動作します。Payara Server 4 のクラスタでは Shoal (または GMS) と呼ばれる技術をベースとしていましたが、Payara Server 5 ではこれらは完全に削除されました。Payara Server 4 のクラスタ構成・管理コマンドは、移行を容易にするため引き続き Payara Server 5 でもサポートされています。しかし、この方法で作成・管理された Payara Server 5 のクラスタはドメイン・データ・グリッドと同じ技術の上で動作し、古い Shoal クラスタは非推奨となりデフォルトでは新しいクラスタ構成には使用されないため、他のデプロイメント・グループのように振る舞います。

進化したクラウド配備

Payara Server 4 の主な欠点の 1 つとして、Hazelcast クラスタ機能が Shoal によって提供される古いクラスタ・モデルに追加される形を採ることが挙げられます。このため、一般的なクラウド配備シナリオ、特にコンテナ技術 (例えば、Docker や Kubernates など) によって支えられている環境において、Hazelcast ク

ラスタ機能はユーザー・フレンドリーではありません。Payara Server 5 には、クラウド環境と統合したより良いクラスタ構成や、クラウド環境におけるほとんどの一般的なユース・ケースをカバーする使いやすい構成オプションなど、いくつかの改良が含まれています。データ・グリッドが提供する探索モードの例には以下のものが含まれます。

- **TCPIP:** IPv4 または IPv6 ネットワーク・アドレスによって識別されるホストの一覧に存在するインスタンスを探索する
- **DNS:** ホスト名によって識別されるホストの一覧に存在するインスタンスを探索する
- **Multicast:** マルチキャスト・プロトコルによりクラスタ内のインスタンスを互いに通信可能にしてそれらを結びつける
- **Kubernetes:** Kubernates クラスタに含まれるホストに存在するインスタンスを探索する

クラウド環境に関する議論の的になる重要な機能の 1 つは**エラスティシティ (elasticity; 弾性)** です。これはクラウド環境が想定されるユーザー負荷 (およびその他の要因) に合わせてスケールアップまたはスケールダウンする能力をいいます。エラスティシティはほとんどのクラウド環境において最も注目されるもののひとつであり、ドメイン・データ・グリッドは主要な 3 つのクラウド環境上におけるエラスティック・アレンジメント (elastic arrangement) を備えています。Payara Platform でエラスティック・アレンジメントを開発する際に考慮すべき重要なことは、デフォルトで Payara Server および Payara Micro の双方がドメイン・データ・グリッドによる**エラスティック・クラスタリング (elastic clustering)** をサポートしていることです。なお Payara Server はデプロイのグループ化もサポートしますが、グループ化されたデプロイについてはデプロイメント・グループが集中構成を採る特定のインスタンス群を対象としているため、**エラスティシティ**をサポートしません。デプロイメント・グループは集中管理された従来タイプのクラスタの代替としてより適しています (詳細については後述のデータ・グリッドとデプロイメント・グループの使用方法をご覧ください)。一方、Payara Micro はデプロイメント・グループの概念をサポートしません。デプロイされたアプリケーションのライフサイクルはインスタンス自身と一体化しています。これは Payara Micro がエラスティックなクラウド環境に特化した設計となっているためです。

移行手順

準備

Payara Server 5 は **JDK 8 が必須**です。もし Payara Server 4 のドメインが現在 JDK 7 で動作しているのであれば、移行を開始する前に JDK のアップデートが必要です。

また、以下の点についてもご注意ください。

- Payara Server 5 は Java EE アプリケーションをサポートします。Payara Server 4 を移行する際に、アプリケーションで JSON と Java オブジェクトのシリアライズを使用している場合には注意が必要となります。Java EE 8 には新しい JSON-B API が同梱されており、既存のアプリケーションで不

具合が発生する可能性があるためです。加えて、Servlet API (4.0) の新機能として HTTP/2 サポートが導入されました。これら 2 つの詳細については後述のセクションでご説明します。

- Payara Server 5 は MicroProfile 2.0 もサポートします。これは Java EE 8 と互換性のある API です。デプロイするアプリケーションが MicroProfile API の機能に依存している場合、非互換性についてはまったく心配する必要はありません。

> このガイドの執筆時点では、バージョン 8 よりも新しい JDK のサポートは実装されていません。JDK 11 の公式なサポートは今後の Payara Server 5 で導入する予定です。現時点では 2019 年の第 2 四半期または第 3 四半期を目標にしていますが、潜在している解決すべき技術的な課題により変更される可能性があります。

バックアップとリストアを用いた Payara Server 4 からのドメインの移行

運用中の Payara Server 4 ドメインを Payara Server 5 への移行戦略として推奨する方法の 1 つは、ドメインのバックアップを実行して、それを Payara Server 5 でリストアする方法です。注意していただきたいのは、この戦略は現在の構成をそのまま Payara Server 5 にインポートする方法であるため、新機能 (ドメイン・データ・グリッド、H2 database、HTTP/2 プロトコル、等) を利用できるようにするには、以下のセクションに示す構成変更を実施しなければならないことです。

この手順では、各々の環境で以下の手順を実行してください。

1. まず、Payara Server 4 のドメインを停止する必要があります。ドメインのバックアップ・プロセスは当該ドメインが動作していない時に限り実行できます。そのため、現在商用環境にある Payara Server 4 のドメインについて停止時間をスケジューリングする必要があります。
2. asadmin コマンドの backup-domain を実行し、ドメインのバックアップを含む圧縮ファイルを配置するディレクトリのパスを指定します。

```
asadmin> backup-domain --backupDir <path-to-backup-directory> <domain-name>
```

圧縮ファイルは以下の場所に配置されます。

```
<path-to-backup-directory>/<domain-name>/<domain-name>_<yyyy_mm_dd>_v<backup_
number>.zip
```

backup_number プレースホルダーは何回目のドメイン・バックアップであるかを表す連番となります。

3. ドメインの完全なバックアップを取った後、Payara Server 5 でそれをリストアします。以下のコマンドを実行します。

```
asadmin> restore-domain --filename <path-to-backup-directory>/<domain-
name>/<domain-name>_<yyyy_mm_dd>_v<backup_number>.zip --long <domain-name>
```

コマンドはリストア結果の詳細を出力します。

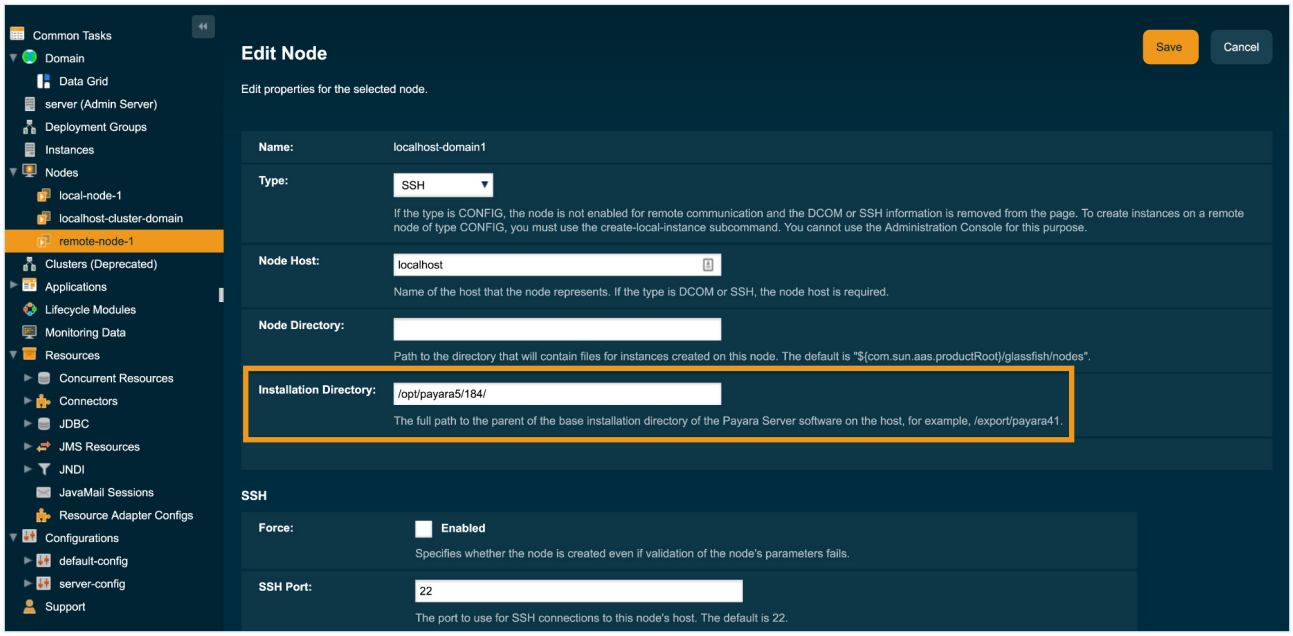
```
Restored the domain (<domain-name>) to /opt/payara5/184/glassfish/
domains/<domain-name>
Description                : <domain-name> backup created on <yyyy_mm_dd> by
user <username>
GlassFish Version          : Payara Server 5.184 #badassfish (build 24)
Backup User                : <username>
Backup Date                : <backup-timestamp>
Domain Name                : <domain-name>
Backup Type                : full
Backup Config Name         :
Backup Filename (origin)   : <path-to-backup-directory>/<domain-name>/<domain-
name>_<yyyy_mm_dd>_v<backup_number>.zip
Domain Directory           : /opt/payara5/184/glassfish/domains/<domain-name>

Command restore-domain executed successfully.
```

4. 最後に、リストアしたドメインを起動します。

ノードとインスタンスに関する追加の考慮事項

もしドメイン構成に別のノード上にあるインスタンス定義が含まれている場合、リストアの際にドメインが完全に動作するよう、以下の推奨事項を考慮する必要があります。

1. Payara Server 5 のバイナリ・ファイルをドメイン構成で定義されているものと同じ場所に手動でインストールする必要があります。この場合、Payara Server 4 のバイナリを置き換える必要があることに注意してください。それは運用中の Payara Server 4 ドメインに先立って行う必要があるということです。これは Payara Server 5 のインストールを中止して Payara Server 4 へ戻る際に問題となり得ます。それを避けるためには、リモート・ホストのインストール・ディレクトリを変更するのが最も良い方法です。リモート・インスタンスを開始する前に、管理コンソールでこの構成を変更してください。



2. これらのインスタンスで作成された、または関連づけられた情報を DAS と再同期する必要があります。これを実行するためには、ドメインの各ノードでインスタンス (ローカル・ノードまたはリモート・ノード) を手動にて起動する必要があります。これらのインスタンスを起動する際には、--sync 引数に full を設定することで、各インスタンスが正しく再構築されます。

```
asadmin> start-local-instance --sync=full <instance-name>
```

Payara Server 5.184 における特別な考慮事項

Payara Server 5.184 では、サーバーを実行する JDK 8 のアップデートに特定の必要条件を導入することで、**Grizzly NPN** フレームワークに含まれるいくつかの SSL 関連クラスによって必要となる変更を回避できるようにしました。このフレームワークは Web コンテナで HTTP/2 プロトコルを使用可能にするもので、フレームワークのバージョンに依存し、同様に JDK のバージョン要求についても厳密に決まっています。もし互換性のない JDK 8 アップデートを Payara Server 5.184 に使用するとサーバーの起動に支障が出ます。これを (将来の互換性の問題も含めて) 解決する最良の方法は、サーバーのドメイン構成を手動でアップデートすることです。

1. ドメイン構成ファイル (***domain.xml***) を開き、`server-config` 構成ツリー内にある以下の JVM 引数の設定箇所を探します。

```
<java-config classpath-suffix="" debug-options="-agentlib:jdwp=transport=dt_
socket,server=y,suspend=n,address=9009" system-classpath="">
   <!-- ... -->
   <jvm-options>-Xbootclasspath/p:${com.sun.aas.installRoot}/lib/grizzly-npn-
bootstrap.jar</jvm-options>
  <!-- ... -->
</java-config>
```

2. 上記 JVM 引数の設定箇所を、以下に示す要素のセットで置き換えます。

```
<java-config classpath-suffix="" debug-options="-agentlib:jdwp=transport=dt_
socket,server=y,suspend=n,address=9009" system-classpath="">
  <!-- ... -->
   <jvm-options>[1.8.0|1.8.0u120]-Xbootclasspath/p:${com.sun.aas.installRoot}/
lib/grizzly-npn-bootstrap-1.6.jar</jvm-options>
   <jvm-options>[1.8.0u121|1.8.0u160]-Xbootclasspath/p:${com.sun.aas.
installRoot}/lib/grizzly-npn-bootstrap-1.7.jar</jvm-options>
   <jvm-options>[1.8.0u161|1.8.0u190]-Xbootclasspath/p:${com.sun.aas.
installRoot}/lib/grizzly-npn-bootstrap-1.8.jar</jvm-options>
   <jvm-options>[1.8.0u191|1.8.0u500]-Xbootclasspath/p:${com.sun.aas.
installRoot}/lib/grizzly-npn-bootstrap-1.8.1.jar</jvm-options>
  <!-- ... -->
</java-config>
```

これにより、移行後のドメインは JDK 8 のアップデートに合わせて互換性が保たれ、ドメインは将来の移行にも備えた形になります。

もしドメインが追加のインスタンスを実行するため複数の構成を持っているならば、同様の変更をそれらの構成に適用する必要があります。

クラスタと高可用性

Payara Server 4 におけるクラスタリング・オプションのまとめ

Payara Server 4 は 2 種類のクラスタリング・メカニズムをサポートしています。Payara Server 5 では Hazelcast ベースのクラスタリング・メカニズムを発展させ、ドメイン構成の中に上手く統合しています。言い換えると、GlassFish から引き継いだ従来のクラスタはまだ存在はするものの、今では非推奨となっています。

Payara Server 4 が最初にサポートしたメカニズムは Shoal クラスタで、これは Payara Server が GlassFish Open Source Server から引き継いだ従来のクラスタリング・メカニズムです。Shoal クラスタはシステマティックな方法で準備する必要があり、新しいインスタンスは手動で追加できるようになっており、また削除についても同様です。このメカニズムは信頼性は高いものの、長年にわたりいくつもの制限が積み重なっていました。

- クラスタの準備に様々なことが必要である: DAS にクラスタを設定し、ローカルまたはリモートの各ノードを設定し、すべてのクラスタ・ホストに対して SSH アクセスを設定する (リモート・ノードの場合)。
- インスタンスは手動でクラスタに追加または削除しなければならないため、クラウド環境においては、スケールアップ・スケールダウンが常に扱いづらくとても面倒な作業となる。
- 特定のデータ (Web セッション・データおよびステートフル・セッション・ビーン) のみがクラスタ全体で複製および保存される。
- インスタンス間での通信の確率に用いられる内部プロトコルは、クラスタの生存期間を通じてパフォーマンス低下の副作用をもたらす。

もうひとつのメカニズムは Hazelcast クラスタです。これは Hazelcast を使用して、新しい Payara Server および Payara Micro インスタンスを必要に応じてクラスタへ追加または削除できるようにカスタマイズされたクラスタを構成できます。この機能は Payara Server 4.1.1.161 から利用可能となっており、Payara Server はクラウド環境上で利用しやすくなり、高可用性環境を素早く構築するためのプロビジョニング作業が簡単になりました。こうしたメリットの一方で、以下のような新たな課題も出てきました。

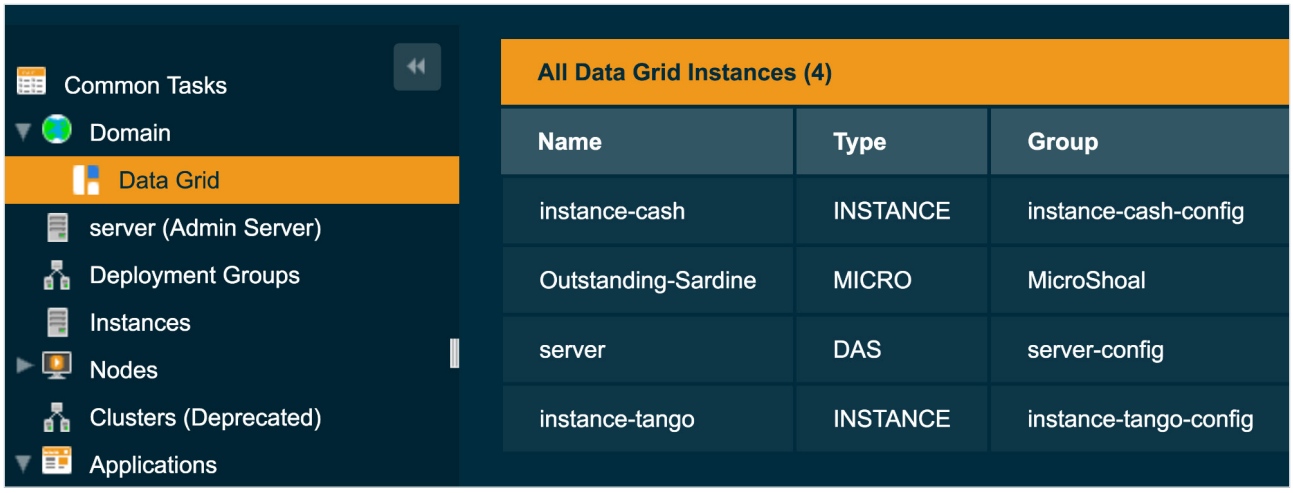
- サーバーのすべてのノード / インスタンスで Hazelcast を手動で有効化する必要があります。
- Hazelcast は新しいインスタンスの探索を自動で行い、検出時にそれらを自動でクラスタに参加させることが可能ですが、複数のクラスタがプロビジョニングされている特定の環境では、セットアップに難があります。
- Hazelcast はクラスタのノード間通信にデフォルトでマルチキャスト・プロトコルを使用します。クラウド・プロバイダーや、コンテナのオーケストレーション・ツールによってはマルチキャスト・プロトコルがサポートされません。そのため、プロビジョニング対象となるすべてのクラスタ・インスタンスについて、他のプロトコルを使用するようにカスタマイズした Hazelcast 構成ファイルを使用しなければならないのです。これはよくある問題です。

Payara Server 5 のドメイン・データ・グリッド

Payara Server 4 のクラスタに関する課題をすべて克服するため、Payara Server 5 では**ドメイン・データ・グリッド**の概念を導入しています。ドメイン・データ・グリッドは、Payara ドメインに属するすべての Payara Server インスタンスに対して、分散型のインメモリ・データ構造を提供します。データ・グリッドはより可用性が高く、より拡張性に富み、ドメインのすべての Payara Server 間でインメモリ・データ・ストレージとレプリケーションを使用可能にします。

ドメイン・データ・グリッドは、Payara Server 4 がサポートする Hazelcast クラスタを進化させたものです。このクラスタからドメイン・データ・グリッドへのアップグレードはシームレスに行うことが可能で、構成の変更を必要としません。Payara Server 4 で Shoal クラスタを使用している場合には、Payara Server 5 でも同じ管理コマンドを用いて同様のクラスタを引き続き管理することができます。Shoal/GMS 技術は使用されませんが、代わりに Hazelcast がその動作を補助し、デプロイメント・グループが同様の振る舞いを実現します。このガイドで後述するクラスタリングの変更点に関するセクションによく注意して、ドメイン・データ・グリッドやデプロイメント・グループへの移行関連を計画してください。

Payara Server 4 では Hazelcast を明示的に有効化する必要があったのに対し (例えば、JCache API を使用する場合や、Hazelcast を Web セッション永続化として使用する場合などは、Hazelcast を有効化する必要があります)、Payara Server 5 では **Hazelcast はデフォルトで有効**となっています。これはすなわち、デフォルトで、 **ドメインに属するすべてのインスタンスが自動的にドメイン・データ・グリッドに参加**し、セッション・レプリケーション、分散キャッシュ、埋め込み Hazelcast グリッドといった機能の恩恵を受けられることを意味しています。ドメイン管理サーバー (DAS) はすべてのインスタンスを把握して、インスタンス間の通信に関するすべてを取り持ちます。DAS はデータ・グリッド上のすべてのインスタンスに関する情報を `asadmin` コマンドや管理コンソール (下図) によって提供することもできます。



ドメイン・データ・グリッドは**実行中のインスタンスのみで構成**されます。デフォルトでは、ドメインで作成されたすべてのインスタンスは起動時にデータ・グリッドに参加します。ドメイン・データ・グリッドは、ドメイン内に構成されていないインスタンスであっても、同じデータ・グリッドに接続するように構成されていればそれも含めることができます。

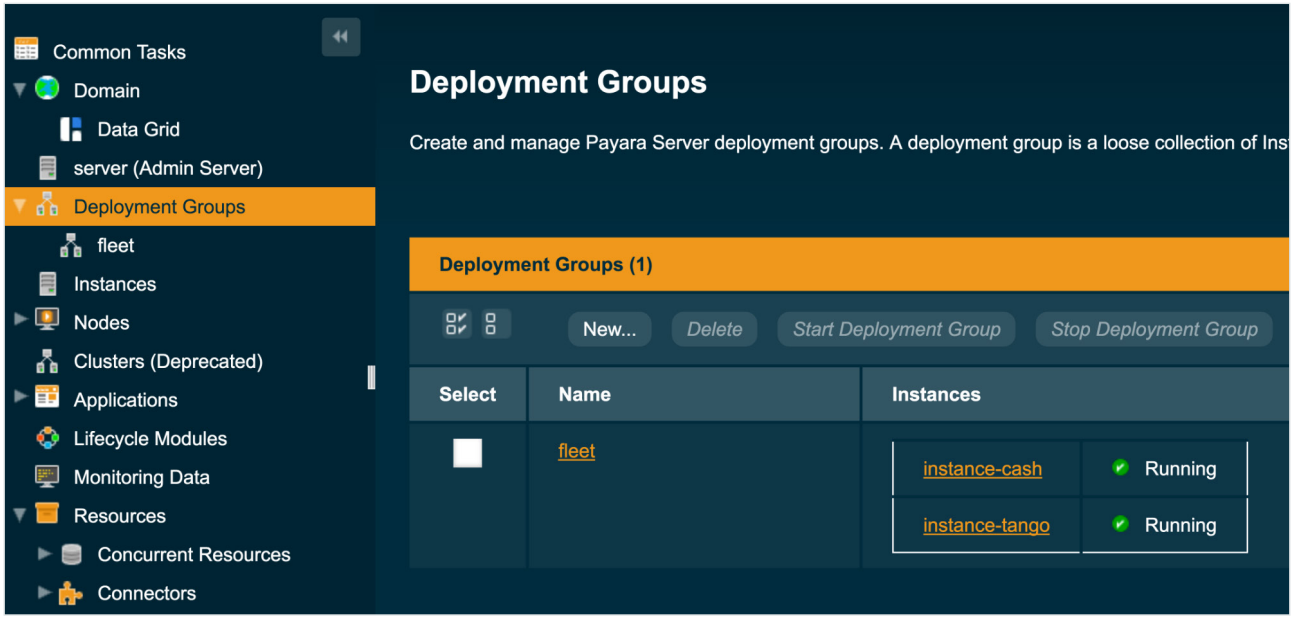
データ・グリッドに参加するインスタンスは以下に分類されます。

- **DAS:** ドメイン管理サーバー自身
- **INSTANCE:** DAS のように同じドメインの一部となっている個々のインスタンス (例: `instance-tango`)、または他のドメインから参加しているインスタンス (例: `instance-cash`)
- **MICRO:** ドメイン・グリッドに明示的に接続している Payara Micro インスタンス

Payara Server 5 のデプロイメント・グループ

ドメイン・データ・グリッドはとても柔軟で、ドメイン内にひとつのグリッドだけ存在します。ドメイン・データ・グリッドに紐付くすべてのリソースは、データ・グリッドのすべてのインスタンスで共有されます。さらに、データ・グリッドの各インスタンスは個別に管理され、アプリケーションは各インスタンスへ個別にデプロイすることもできます。これ自体は特に問題はなく、動的にスケールする環境では望ましいものでさえあります。しかし、ドメイン・データ・グリッド自身は以前の Shoal クラスタ・モデルが有している機能のすべてを提供できるわけではありません。そこで Payara Server 5 で導入されたものが、デプロイメント・グループという概念です。

デプロイメント・グループはドメイン・データ・グリッドの機能を拡張する形で動作します。デプロイメント・グループはインスタンスの管理された集合体であり、アプリケーションやリソースを共有します。このインスタンスの集合体はデータ・グリッドの機能拡張として負荷分散やフェール・オーバーを提供し、以前の Shoal クラスタと同じような働きをします。



デプロイメント・グループの構成例を上図に示します。インスタンス *instance-tango* と *instance-cach* は *fleet* というデプロイメント・グループに属しており、*solo-instance* という第 3 のインスタンスはグループに属しておらず別個に管理されます。

スタンドアロン・インスタンス

Payara Server 4 には異なる 2 種類のインスタンスがあります。

- クラスタ・インスタンスは、クラスタ以下に直接作成されたインスタンスで、クラスタ単位で独立しており、そのライフサイクルはクラスタに直接紐付けられます。
- スタンドアロン・インスタンスはクラスタに含まれないインスタンスです。スタンドアロン・インスタンスは互いに完全に独立しており、リソースもアプリケーションも共有しません。各スタンドアロン・インスタンスは個別に管理されます。

一方 Payara Server 5 では、ドメイン・データ・グリッドとデプロイメント・グループという概念においてインスタンスに明確な区別はありません。このモデルでは、作成されるすべてのインスタンスは管理上スタンドアロン・インスタンスとして扱われます。つまり、Payara Server 4 においてインスタンスのライフサイクル管理のために使用する管理コマンド (`create-instance`、 `delete-instance`、等) を、Payara Server 5 上でも同じように使用するということです。主な違いは、Payara Server 5 のインスタンスは自動的にドメイン・データ・グリッドに参加し、必要に応じてデプロイメント・グループに追加できる点にあります。Payara Server 4 では、スタンドアロン・インスタンスは従来の Shoal クラスタに追加することはできず、構成を修正することにより Hazelcast クラスタに参加することができます。クラスタ・インスタンスは Payara Server 5 でも後方互換性のため古いクラスタ・モデルの一部として残っていますが、その動作はデプロイメント・グループでグループ化された他のスタンドアロン・インスタンスと非常によく似たものとなっています。

この違いは、Payara Server 4 でスタンドアロン・インスタンスを使用している場合には明確に意識しておく必要があります。ドメインを Payara Server 5 へ移行する際、これらのインスタンスは問題なく動作しますが、自動的にドメイン・データ・グリッドにも参加するため、実質的には Payara Server 4 における Hazelcast クラスタのインスタンスのように動作します。

Payara Server 5 のクラスタ・オプションのまとめ

	ドメイン・データ・グリッド	デプロイメント・グループ	クラスタ (非推奨)
Hazelcast ベース	✓	✓	✓
実行中のメンバー・インスタンスをドメイン・データ・グリッドで表示可能	✓	✓	✓
DAS からのみ管理	✗	✓	✓
デプロイ対象にすることが可能	✗	✓	✓
メンバー・インスタンスを同時に起動/停止可能	✗	✓	✓
負荷分散とフェール・オーバーをサポート	✗	✓	✓
インスタンスごとに異なる構成が可能	✓ *1	✓	✗
メンバー・インスタンスをインスタンス一覧で表示可能	✓	✓	✗
インスタンスを (クラスタを構成せず) 動的に接続可能	✓	✗	✗
Payara Server 4 のクラスタ管理コマンドとの互換性	✗	✗	✓
Payara Micro インスタンスを参加させることが可能	✓ *2	✗	✗

1) 同一ドメイン内に限る
2) Payara Micro を Payara Server と同じ探索モードで起動した場合 (デフォルトでは異なる探索モードを使用する)

以下は**ドメイン・データ・グリッド**に参加できます。

- デプロイメント・グループに属するインスタンス
- クラスタ (非推奨) で作成したインスタンス
- 他のドメインのインスタンスだが同じデータ・グリッド・グループに参加できるよう構成したもの
- Payara Micro インスタンス

以下は**デプロイメント・グループ**に参加できます。

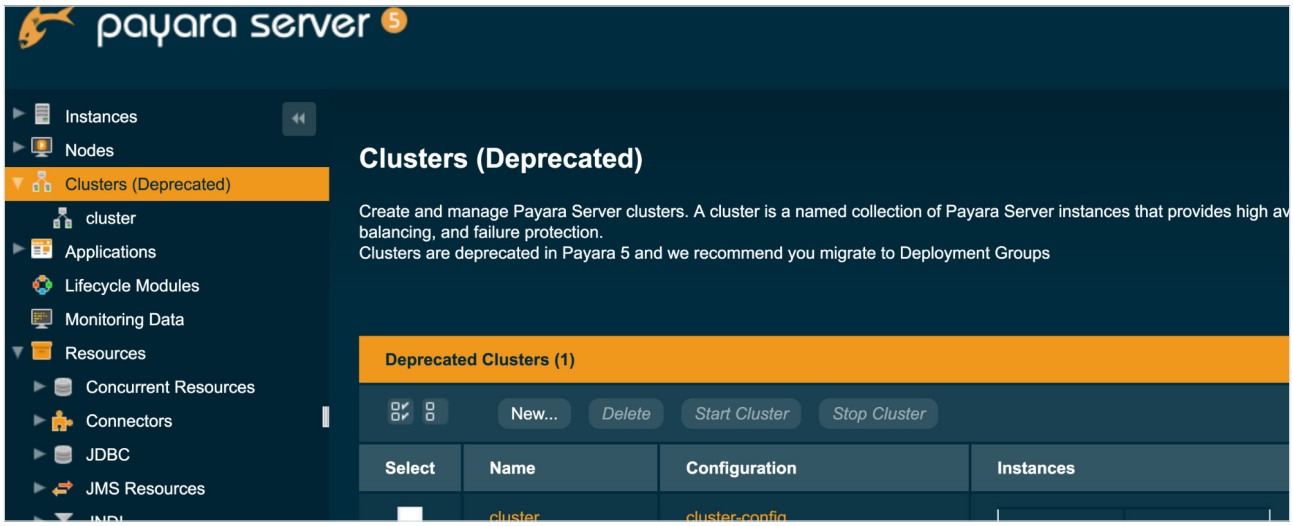
- デプロイメント・グループと同時に作成したインスタンス
- デプロイメント・グループとは別に作成し、後からグループに追加されたインスタンス

以下は**クラスタ (非推奨)** に参加できます。

- クラスタ専用に作成したインスタンス

Payara Server 4 標準クラスタの維持

前述の通り、Payara Server 5 は Payara Serve 4 から移行した Shoal クラスタの構成を解釈することができます。もしドメインに非推奨のクラスタが含まれていても、変更することなく Payara Server 5 の同一ドメインで起動することが可能です。主な相違点は、Payara Server 5 ではクラスタのプロビジョニングを行うため Shoal/GMS 技術に代えて Hazelcast グリッドを使用することです。また、従前の機能を持つクラスタは、管理コンソール上では「**Cluster (Depricated)**」として区別されます。



これは Payara Server 4 からの移行を行いやすくするための、Payara Server 5 の便利な機能です。従前のクラスタは、現在では上記の画像のように管理コンソールの「**Cluster (Deprecated)**」ビューで管理されます。

> ***デプロイメント・グループへ移行してください***
>
> Payara Server 4 のクラスタは Payara Server 5 でも動作しますが、このクラスタは非推奨となっているため、**デプロイメント・グループ**への移行をお勧めします。移行方法については以下のセクションをご参照ください。

Payara Server 4 標準クラスタからデプロイメント・グループへの移行

デプロイメント・グループへの移行を決めたのであれば、クラスタ管理により柔軟な方法を採れるようになります。デプロイメント・グループはクラスタと似ていますが、さらに個々のクラスタに特化したインスタンスを作成する必要はなく、インスタンスの作成や構成を別個に行い、後からデプロイメント・グループへ追加または削除することができます。ひとつのインスタンスを複数のデプロイメント・グループに追加することもできます。

Payara Server 4 のクラスタは、Payara Server 5 へのアップデート時に直接デプロイメント・グループに移行することができます。あるいは、アップグレード時にはクラスタを維持しつつ (前のセクションにてご説明しました)、後から Payara Server 5 の非推奨クラスタからデプロイメント・グループへと移行することもできます。移行計画に合わせて最適な方法をご選択ください。

構成を維持しつつ移行前のクラスタとできるかぎり同じように動作させる場合は、以下の手順に従ってください。

1. カスタムのクラスタ構成をコピーします
    - もしクラスタがカスタム構成を含む場合、クラスタ関連の構成 (例: cluster-config) をコピーしておきます。
    - もしくは、その内容をメモして、後で新しいデプロイメント・グループに適用します。
2. すべてのクラスタ・インスタンスをスタンドアロン・インスタンスに変換します
    - ドメインを停止させ、DAS の domain.xml ファイルを手動で編集し、<clusters> 要素をすべての子要素を含めすべて削除します。
    - 上記はすべてのクラスタを削除する手順です。もし複数のクラスタを構成している場合には、<clusters> 要素に含まれる、移行対象とするクラスタに関連する <cluster> 要素のみを削除します。
3. デプロイメント・グループを作成します
    - ドメインを開始します。
    - デプロイメント・グループを作成します (クラスタが既に存在しなければ、その名前に移行元のクラスタ名を用いることができます)。
    - 移行前のクラスタに属していたインスタンスを、新しいデプロイメント・グループに追加します。
    - 必要に応じて、デプロイメント・グループにカスタム・リソースを作成します (以前のクラスタ構成をコピーしなかった場合)。

新しいデプロイメント・グループでは、以前のクラスタで実行していたものと類似した操作を実行することが可能です。以下にそれぞれの操作の対応表を示します。

操作	デプロイメント・グループ		クラスタ	
	管理コンソール	asadmin コマンド	管理コンソール	asadmin コマンド
すべてのインスタンスの起動	Start Deployment Group	`start-deployment-group`	Start Cluster	`start-cluster`
すべてのインスタンスの停止	Stop Deployment Group	`stop-deployment-group`	Stop Cluster	`stop-cluster`

インスタンスの作成	New instance in the group.
	
	And existing instance to the group
		`create-instance`
		
		`add-instance-to-deployment-group`
			New instance in the cluster	`create-instance --cluster`

クラスタで使用できる構成のいくつかはデプロイメント・グループでも使用できます。例えば、デプロイされたアプリケーション、リソース、プロパティといった構成は、グループ内の各サーバーの構成の上に適用されます。デプロイメント・グループで設定していないその他すべての構成については、個々のインスタンス向けに編集した構成か、もしくは以下のものが適用されます。

- バッチ構成は各構成の Batch ページ (管理コンソールのサイドバー) のものが適用されます。
- JMS 物理宛先については、現在の Payara Server では構成できなくなっています。代わりに、mq/bin ディレクトリの imqadmin または imqcmd ツールを使用し、デプロイメント・グループが使用する MQ サーバーに直接接続してください。デプロイメント・グループ全体に JMS リソースを追加するには、リソースの対象にデプロイメント・グループを追加してください。

Hazelcast クラスタのスタンドアロン・インスタンスからドメイン・データ・グリッドへの移行

Payara Server 4 では、Hazelcast のメカニズムをベースとしたクラスタを構築する方法として、単一のクラスタ構成を参照する複数のスタンドアロン・インスタンスを作成して相互に接続する方法を推奨していました。この方法を用いているのであれば、このセクションでご説明するドメイン・データ・グリッドへの移行方法は、Payara Server 5 のセットアップに似たものとなります。一度この作業を行うことで、デプロイメント・グループでグループ化したインスタンスは単一のクラスタのように管理することができます。これは Payara Server 4 では実現できなかったことです。

Payara Server 4 のスタンドアロン・インスタンスを移行する場合には、構成を変更する必要はありません。既存の Payara Server 4 ドメインをこのように構成している場合には、構成を変更することなくそのまま Payara Server 5 で使用することができます。以下に示す変更点を除いて、ドメイン・データ・グリッド上にインスタンスが存在し、Payara Server 4 と同様に動作させることができます。

- Hazelcast の探索メカニズムがマルチキャストから、Payara Server 5 で新たに導入された domain 探索モードに変更されています。このモードでは、ドメイン管理サーバー (DAS) はいずれのインスタンスに接続する必要もなく、代わりにインスタンスが DAS と「対話」することでデータ・グリッドに参加します。そのため、DAS が他のインスタンスよりも先に起動するか、あるいは各インスタンスは DAS がデータ・グリッドに接続するまで 5 分以内の待機時間を要します。この時間は Hazelcast の merge delay システム・プロパティを変更することにより短くすることが可能です。探索モードには従来のように multicast 探索を使用することもできます。その場合は、管理コンソールのデータ・グリッド構成または asadmin コマンドの `set-hazelcast-configuration --clustermode multicast` を使用してください。
- DAS はサーバー・インスタンスとは異なる Hazelcast ポートをリスンします。デフォルトでは、DAS はポート **4900** をリスンし、他のインスタンスは **5900** で始まるポートをリスンします。Payara Server 4 では、DAS とインスタンスはデフォルトで **5900** から始まるポートを使用します。
- Payara Micro インスタンスは、デフォルトの構成では Payara Server 5 のデータ・グリッドに自動的に接続することはできません。

Payara Micro インスタンスのクラスタの維持

複数の Payara Micro インスタンスでクラスタを構築している場合、Payara Micro 5 では構成の変更なく実行できます。 **Payara Micro 5 の探索メカニズムはマルチキャストのままである**ため、インスタンスは今まで通りに探索およびクラスタの構築が可能です。ただし、以下の点にご注意ください。

- デフォルトの開始ポートが **5900** から **6900** に変更となっています。これを従前に戻すには、各インスタンスを開始する際に `--startport` コマンドライン引数を使用してください。
- デフォルトのマルチキャスト・アドレスとマルチキャスト探索のポート番号は Payara Server と Payara Micro で異なっています。Payara Micro でこれらの設定を変更するには、各インスタンスを開始する際に `--mcport` および `--mcaddress` コマンドライン引数を使用してください。
  - Payara Server では、デフォルトの mcaddress (マルチキャスト・グループ) は 224.2.2.3 で、mcport (マルチキャスト・ポート) は 54327 です。Payara Server 側で変更しない場合には、Payara Micro 実行時にこれらを指定してください。
- 現在は `--clustermode` コマンドライン引数を使用することで、マルチキャスト以外の探索メカニズムを簡単に構成できるようになっています。Payara Server 4 では、カスタマイズした hazelcast.xml 構成ファイルによって手動で各探索モードを構成する必要がありました。

Payara Server および Payara Micro インスタンスの混成クラスタの維持

Payara Micro 4 のインスタンスは自動的に Payara Server 4 インスタンスのクラスタに参加することが可能で、クラスタに参加しているインスタンスの一覧は DAS 上から確認することができます。これは通常 Payara ハイブリッド・クラスタとして知られています。一方で、前のセクションでもご説明したとおり、Payara Micro 5 インスタンスは Payara Server のドメイン・データ・グリッドに自動的には参加することはません。Payara Platform でハイブリッド・クラスタを移行する際には、以下の 2 つの選択肢を検討してください。

- Payara Server のドメイン・データ・グリッドを `multicast` 探索モードを使用するよう構成して、クラスタの動作を Payara Server 4 と同様にする
- Payara Micro インスタンスで `domain` 探索モードを使用して、ドメイン・データ・グリッドに接続する

> Payara Server および Payara Micro インスタンスのアップグレードは一度に同じバージョンへと行う必要があります。仮にそうしなかった場合には、思わぬ副作用が発生して、クラスタ内のデータが失われる可能性があります。

ほとんどのケースでは第一選択肢として、新しい `domain` 探索モードを推奨します。`domain` 探索モードはより強力で、どのようなネットワーク構成でも動作し、クラウド環境では特に適しています。この探索モードは DAS の IP アドレスまたはホスト名のみが必要となります。`domain` 探索モードを使用するには、Payara Server 5 ドメインを起動し、以下に示すような DAS のログが出力されることが確認され、ドメイン・データ・グリッドが完全に起動するまで待機します。

```
Info:   Data Grid Status
Payara Data Grid State: DG Version: 35 DG Size: 1
Instances: {
Address: /192.168.1.51:4900 UUID: f5ce0097-77b4-4904-8f30-a2fd7a121fb8 Lite:
false This: true Name: server Group: server-config
}
```

その後、Payara Micro 5 インスタンスを以下のように起動します。

```
java -jar payara-micro-5.jar --clustermode domain:<das-hostname>:<cluster-port>
```

このようにすることで Payara Micro インスタンスはデータ・グリッドに参加し、データ・グリッドの状態が DAS のログもしくは管理コンソールに表示されます。

> Payara Micro インスタンスを起動するときに、DAS が動作している必要はありません。インスタンスはすぐにはクラスタに参加せず、DAS がクラスタに接続可能になるまで待機します。ただし、DAS が起動してそれを Payara Micro が検出するまで 5 分以上かかる場合もあることから、Payara Micro インスタンスの起動時には DAS は動作させておいた方が良いでしょう。

Payara Server 4 同様のマルチキャスト探索を引き続き使用する場合は、Payara Server 5 のデータ・グリッド探索モードを `multicast` に変更する必要があります。また、Payara Micro 実行時に `--mcport` および `--mcaddress` コマンドライン引数で Payara Server と同じマルチキャスト・アドレスとポートを設定する必要もあります (デフォルト値が異なるため)。この方法は、データ・グリッドに参加する他の Payara Server 5 インスタンスに影響することにご注意ください。この選択肢を用いる場合は、すべての Payara Server または Payara Micro インスタンスについて細心の注意を払ってください。

JSON および Java オブジェクトのマッピング

JSON マッピングの変更について

Payara Server 5 では Java EE 8 アプリケーションを実行することができます。新しいバージョンの Java EE 標準を使用するメリットの 1 つとして、JSON-B (JSON Binding) API が含まれていることが挙げられます。この API は POJO のシリアライゼーションを JSON のペイロードとして定義したり、その逆を行うために用いることができます。この新しい API は、代替手段としてよく用いられてきた Jackson ライブラリの影響を受けています。Payara Server 5 の JSON-B が持つ主な長所の 1 つとして、追加設定なしで JAX-RS コンテナと連携することが挙げられます。JSON-B は JAX-RS REST サービスのリクエスト・レスポンス双方のペイロード管理の 1 つである POJO の自動シリアライゼーション・デシリアライゼーションとして使用されます。すべては標準の JSON-B マッピングによって定義され、カスタム拡張によって提供される非標準のマッピングに依存しません。

Payara Server 4 でも JAX-RS コンテナで Java オブジェクトと JSON の自動マッピングを行う仕組みが提供されていますが、Payara Server 5 と比較すると、このマッピングは JAX-B (Java XML Binding) API によって提供されており、Java オブジェクトと XML のマッピングに合わせて設計されていることから、JSON 形式では使いやすいものとは言えませんでした。さらに、このマッピングは XML ペイロードのために標準化されたものであり、他のアプリケーション・サーバーでは基本的に JSON ペイロード向けとしてサポートされているものではありません。Payara Server 4 では、JAX-RS (Jersey) 向けの JSON シリアライゼーション・プロバイダーのデフォルト実装は、EclipseLink コンポーネントが提供する MOXy と呼ばれるライブラリです。Jackson は Payara Server 4 と非常に相性の良い JAX-RS マッパーを提供しており、しばしば Moxy の代わりとして用いられます。

アプリケーションを Payara Server 4 から 5 へ移行する際、もしアプリケーションが JAX-B アノテーションによる自動シリアライゼーションおよびマーシャリング・メカニズムに依存した JAX-RS コンポーネントを宣言している場合、Payara Server 5 ではデフォルトで**これらのアノテーションが無視される**ことに注意

してください。これは JAX-RS の JSON ペイロードのデフォルト・プロバイダーが JAX-B から JSON-B へと変更されたためです。

JAX-B マッピングから JSON-B マッピングへの移行

JSON-B は Java オブジェクトを JSON ペイロードにマッピングする標準的な方法であるため、Payara Server 5 で新しいアプリケーションを作成する場合には最良の選択肢となります。Payara Server 4 からアプリケーションを移行する場合には、すべての JAX-B マッピングをリファクタリングすることは不便かもしれませんが、それでもまずはこの選択肢を検討することをお勧めします。JSON-B を使用するようにリファクタリングすることで、高度に統合されより優れた JSON マッピングをサポートする API を使用することによるメリットが得られるだけでなく、その振る舞いがはっきりしている標準 API をアプリケーションで使用することにより、将来より新しいバージョンの Payara Server にアップグレードする、あるいは他のサーバーへ移行する場合であってもその互換性が保証されます。

現在 JAX-B を用いてアプリケーションのエンティティの JSON のシリアライゼーションとマーシャリングを行っており、それを JSON-B を用いて同一の構成を維持したい場合には、JSON-B 関連のアノテーションを使用するようコードをリファクタリングする必要があります。

Payara Server 4 で JAX-B API による JSON シリアライゼーション (デフォルトでは MOXy プロバイダーを使用します) を構成しているアプリケーションはで、以下のようなアノテーションをクラスに付加していると思います。

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "payload")
public class MyPayload {

    @XmlElement(name = "payloadID", required = true)
    private String id;

    @XmlAttribute(name = "editable")
    private Boolean editable;

    ...
}
```

同じエンティティを Payara Server 5 の JSON-B を用いて構成した例がこちらです。

```
public class MyPayload {

    @JsonbProperty(value="payloadID", nillable=false)
    private String id;
```

```
    private Boolean editable;

    ...
}
```

同じエンティティでもJSON-Bアノテーションを使用した方が理解しやすく表現もシンプルであることがお分かりいただけるかと思います。JSON-B APIの入門としては、公式の Getting started with JSON Binding guide をご覧ください。

JAX-B アノテーションを新しい JSON-B アノテーションと併用することも可能です。以下の場合にお勧めします。

- 同じエンティティを XML と JSON の双方とマッピングさせる場合
- 新しい JSON-B マッピングがどのように働くか JAX-B マッピングと比較したい場合
- 将来 Payara Server 5 で JAX-B マッピングを使用する方法にすぐ戻せるようにしたい場合

Payara Server 5 で JAX-B による JSON マッピングを維持する

Javaクラスと JSON ペイロードのマッピング定義をリファクタリングするために大変な労力を必要とする場面もあるかと思います。そうしたケースでは、Payara Server 4 の場合と同様に JAX-B アノテーションを引き続き使用することもできます。JSON マッピングに JAX-B のアノテーションを使用する場合には、web.xml デプロイメント記述子で以下の Servlet コンテキスト・パラメータを追加してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.
jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1" metadata-complete="false">
  <context-param>
    <param-name>jersey.config.jsonFeature</param-name>
    <param-value>MoxyJsonFeature</param-value>
  </context-param>
    ...
</web-app>
```

この構成では、アプリケーションの JAX-RS サービスは Payara Server 4 と同様に JAX-B アノテーションによるマッピングをサポートします。加えて、JSON-B アノテーションが同じアプリケーション内で使用されていてもそれらは無視されます。ただし、将来的には JAX-B アノテーションのサポートは完全に削除され

る可能性があるため、代わりに JSON-B アノテーションを使用するようアプリケーションをリファクタリングする計画を立てることが現時点で一番望ましいことであることはご理解ください。

Payara Server 5 で Jackson 2 ライブラリによる JSON マッピングを維持する

JAX-RS のエンドポイントで Java クラスから JSON ペイロードへのマッピングに Jackson 2 ライブラリを使用している場合は、Payara Server 5 でも引き続き使用することはできます。長期的に見れば、高度に統合されより優れた JSON マッピングをサポートされ、その振る舞いがはっきりしていてより新しい Payara Server へのアップグレード (あるいは他の Java EE 8 アプリケーションが動作するサーバーへの移行) しても確実に動作する API を使用するほうが望ましいため、JSON-B API を使用するように移行することをお勧めします。加えて、Jackson への依存を削除することでアプリケーションの構成がよりシンプルになります。

Payara Server 5 で Jackson を引き続き使用するには、web.xml デプロイメント記述子に以下の Servlet コンテキスト・パラメータを指定してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.
jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1" metadata-complete="false">
  <context-param>
    <param-name>jersey.config.jsonFeature</param-name>
    <param-value>JacksonFeature</param-value>
  </context-param>
     ...
</web-app>
```

また、アプリケーションには Jackson 2 が必要とするすべての依存ライブラリを含めてください。これは Payara Server 4 の場合でも同様で、Jackson の依存ライブラリはアプリケーションに含まれているはずですので、それ以上の手順は不要です。

この構成では、アプリケーションの JAX-RS サービスは Payara Server 4 でサポートされる Jackson 2 アノテーションと同じマッピングをサポートします。加えて、同じアプリケーションに含まれる JSON-B アノテーションはすべて無視されます。この構成は前のセクションでご説明した JAX-B マッピングの有効化と非常によく似ています。

組み込みデータベース

Payara Server 5 は **H2 database** を同梱しています。これは Payara Server 4 には含まれていません。H2 database は Payara Server 4 の Derby DB に代えてアプリケーションのデフォルト JDBC データソースとして使用されます。Derby DB のデータソースは Payara Server 5 にも存在し、引き続き永続 EJB タイマーのデフォルト・データベースとして使用されますが、Derby DB の使用は**非推奨**となり、将来的には H2 DB で完全に置き換えられます。

組み込みデータベースの変更について

Payara Server の内部機能のいくつか—標準 API セットの一部として現れている機能か、Payara Server 固有機能のいずれかになります—は、サーバーのシャットダウン後もデータを保持する必要があるためデータ・ストアを必要とします。Payara Server 4 では Derby database (JavaDB としても知られています) を同梱し、これらの機能を実現するために使用されていました。このデータベースは 2 つの JDBC リソースの組として現れています。

- `__TimerPool` と呼ばれる埋め込み Derby データベースに接続する接続プール。これは埋め込みデータベースのため、サーバーはこのデータベースを DAS で使用するものと同じ JVM プロセスの内部で起動します。このプールは `jdbc/__TimerPool` JDBC データソースとして現れていますが、アプリケーションからは使用されません。
- `DerbyPool` と呼ばれるスタンドアロン Derby データベースに接続する接続プール。これは JDBC データソース `jdbc/__default` に関連づけられています。このデータソースはアプリケーションのデフォルト・データソースとして使用されます。

しかし、Derby DB は商用運用における問題点 (例えば並行更新時における一貫性がなかったり、予期せぬ行ロックが発生するなど) がいくつか存在しており、現在では時代遅れの製品と考えられています。これらの問題を抱えていることから、Payara Server 5 ではより安定したソリューションとして H2 database を採用しこのデータベースを置き換えることにしました。Payara Server 5 における組み込みデータベースに関する変更点の一覧を以下に示します。

- 埋め込み H2 database へのデータベース接続を構成した H2Pool と呼ばれる新しい JDBC 接続プールが存在します
- jdbc/default データソースは DerbyPool に代えて新しい接続プールにリンクしています
- `__TimerPool` は Payara Server 4 と同様に埋め込みDerby DB にリンクしています
- `DerbyPool` 接続プールは、Payara Server 4 同様に外部の Derby DB にリンクしており、JDBC データソース `jdbc/__default` ではなく `jdbc/__derby` として現れます
- asadmin コマンドの `start-database` および `stop-database` は Derby の代わりに H2 database インスタンスのライフサイクルを管理します

> **商用環境**

> Derby DB または H2 DB のいずれも商用利用は推奨しません。これらはアプリケーション開発を容易にするため Payara Server に含まれています。商用環境では、JDBC リソースおよび関連する JDBC 接続プールを構成するために、別個の商用向けリレーショナル・データベースを使用ことをお勧めします。

H2 Database

H2 DB は `${PAYARA_INSTALL_DIR}/h2db` ディレクトリ以下にインストールされています。

スタンドアロンの H2 database を開始するには、以下の asadmin コマンドを使用します。

```
asadmin> start-database
```

H2 database を停止するには、以下の asadmin コマンドを使用します。

```
asadmin> stop-database
```

Derby Database

Derby DB は `${PAYARA_INSTALL_DIR}/javadb` 以下にインストールされています。これは Payara Server 4 と同じ場所です。

スタンドアロンの Derby database を起動するには、以下の asadmin コマンドを使用します。

```
asadmin> start-database --dbtype derby
```

Derby database を停止するには、以下の asadmin コマンドを使用します。

```
asadmin> stop-database --dbtype derby
```

Payara Server 4 のデータソース構成の維持

Payara Server 5 へのアップグレード時に Payara Server 4 のドメイン構成をそのままコピーした場合は、データソースと接続プールの設定は Payara Server 5 でも同様に動作します。接続プールを動作させ、アプリケーションからアクセスするのに、特別な変更を加える必要はありません。Payara Server 5 では `__TimerPool` と `DerbyPool` 接続プールを引き続き問題なく使用することができます。

> 先に述べたように、Derby DB の使用は***非推奨***となっており、将来的には H2 database に置き換えられる可能性があります。加えて、環境データストアを必要とする新機能については、同様に H2 の使用を前提に開発する予定です。これらの理由から、Derby を有効化する代わりに H2 を使用するように構成を変更することをお勧めします。

Payara Server 5 の新しいデータソース構成への移行

ドメインのアップデート時に Payara Server 5 のデフォルト・データベース構成を使用する場合は、このセクションの方法を採ります。ドメインがデフォルトのデータソースを使用している場合には、アプリケーションが Derby DB の代わりに H2 DB を使用しても動作することを確認してください。デフォルト・データベース構成を使用するには、以下の手順に従ってください。

1. 最初に、以下の asadmin コマンドを実行して `H2Pool` JDBC 接続プールを作成します。

```
asadmin> create-jdbc-connection-pool --datasourceClassname=org.
h2.jdbcx.JdbcDataSource --resType=javax.sql.DataSource
--isIsolationLevelGuaranteed=false H2Pool
asadmin> set 'resources.jdbc-connection-pool.H2Pool.property.URL=jdbc:h2:${com.
sun.aas.instanceRoot}/lib/databases/embedded_default;AUTO_SERVER=TRUE'
```

2. 次に、jdbc/__default データソースを接続プールにリンクします。

```
asadmin> set resources.jdbc-resource.jdbc/__default.pool-name=H2Pool
```

内部データベースを Derby から H2 に変更すると、古いデータベースに保存されたデータは新しい状態のサーバーからアクセスできなくなることに注意してください。Payara Server では今のところ、これらの内部データベースはパーシステント・タイマーの情報 (タイマーはサーバーのシャットダウンや不測のクラッシュにも対応できなければならない) と実行済みバッチ・ジョブの履歴情報に使用しているため、ほとんどのケースでは問題は発生しないと思われます。パーシステント・タイマーの情報は移行のコントロールで問題なくスキップすることができます (データベースが空の場合サーバーは新しいデータを作成します)。そのため、関連のデータセットで役立つものはバッチ・ジョブの履歴データのみとなります。これらのデータを保持したい場合は、データを SQL データ操作スクリプトの形式でエクスポートして、H2

database に挿入することをお勧めします。Derby と H2 は SQL の構文が似ていますので、それほど手間はかからないでしょう。

HTTP/2 プロトコルのサポート

Payara Server 5 は新しい Servlet 4.0 API の一部として HTTP/2 プロトコルのサポートを導入しています。このプロトコルは、サーバー構成に含まれるデフォルトの HTTPS ネットワーク・リスナーにおいてデフォルトで有効です。

HTTP/2 プロトコルに関する変更点

Payara Server 5 では、HTTP/2 プロトコルのサポートはセキュアな HTTP ネットワーク・リスナーにおいて**デフォルト**で有効です。このバージョンの HTTP プロトコルは以下に示すような多くのパフォーマンスの向上をもたらします。

- 必要なコネクション数を削減するためリクエストとレスポンスを多重化
- データ量削減のためヘッダーの圧縮
- 複数の関連するファイルを早く送信するためのサーバー・プッシュ
- セキュリティを向上させるためのコマンドのバイナリ・エンコーディング

加えて、このプロトコルは進化したバージョンの Transport Layer Security (TLS1.2) による暗号化を必要とします。そのため、Payara Server ではセキュアな HTTP ネットワーク・リスナーのみ有効になります。

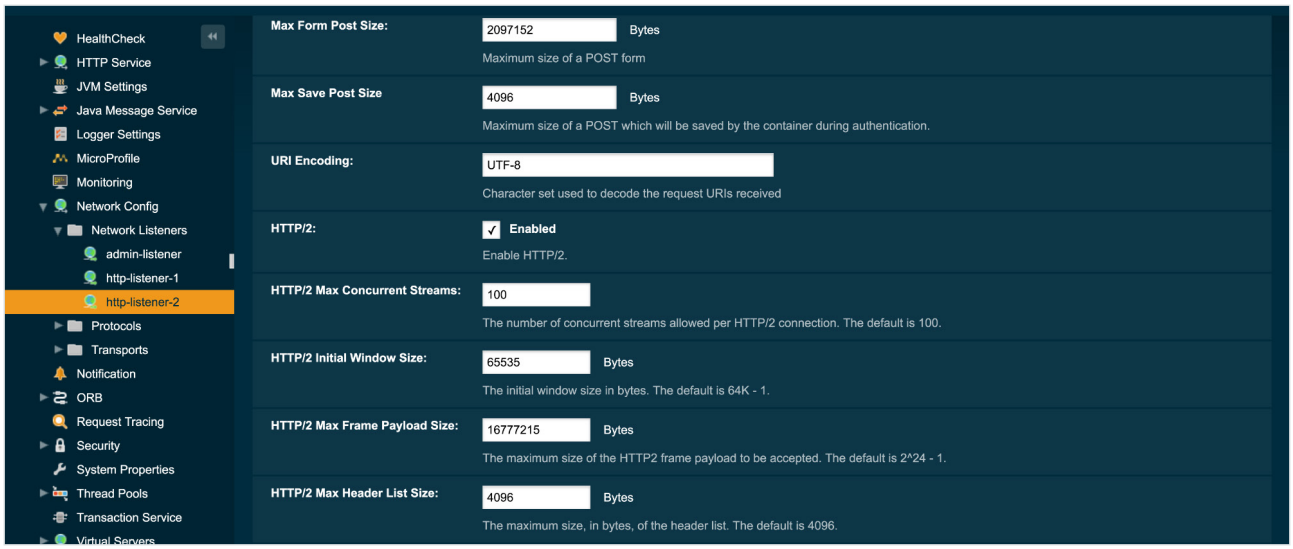
クライアント・アプリケーションで用いられるいくつかの Web フレームワークや Web ブラウザーは HTTP/2 の機能を活用してネットワークのパフォーマンスを最大限に活用しますが、HTTP/2 のサポートはすべてのケースにおいて安定して動作するわけではないため、ご利用のアプリケーションで問題が発生することもあり得ます。そのため、既存のアプリケーションを Payara Server 4 から Payara Server 5 へアップグレードする際には、まずすべての HTTP リスナーで HTTP/2 サポートを無効化して、望ましくないエラーが発生しないようにすることをお勧めします。アプリケーションが Payara Server 上で問題なく動作するようになってから、HTTP/2 を有効にして新たな問題が発生しないかテストを行ってください。

設計上は、HTTP/2 はクライアント証明書による認証をサポートしません。HTTP/2 では、クライアントは複数のアウトスタンディング・リクエストを発行することができます。ある種の関連情報がないため、クライアントはどのリクエストが証明書を要求するサーバーのものであるかを識別することができません。もし認証にクライアント証明書を使用する必要があるのであれば、HTTP/2 を無効化しなければなりません。

すべてのリスナーで HTTP 1.1 を維持する

Payara Server 4 から Payara Server 5 にアップグレードする最も安全な方法は、すべての HTTP リスナーで HTTP 1.1 プロトコルを使用するよう構成には一切の変更を加えず、HTTP/2 を完全に無効化することです。

管理コンソールを使用する場合は、ネットワーク・リスナーのHTTP/2 プロトコルを Network Config → Protocolsオプションで無効化することができます。リスナーを選択した後、HTTPタブに進み、HTTP/2オプションのチェックを外してください。



同様の操作をコマンドラインで実行するには、以下の asadmin コマンドを実行してネットワーク・リスナーの HTTP/2 プロトコルを無効化することができます。

```
asadmin> set configs.config.server-config.network-config.protocols.
protocol.<listener-name>.http.http2-enabled=false
```

移行後の既知の問題

Payara Server 5 への移行が完了した後、現時点でいくつかの問題が発生し得ることが判明しています。下表にその原因と推奨する対応方法について示します。

問題	原因	対応方法
管理コンソール上で DAS 以外のインスタンスのモニター・データを取得できない	管理コンソールが使用する Jersey のコンポーネントの内部的な変更に起因します。	すべての関連するインスタンスのモニタリング・データにアクセスするには、DAS によって提供される REST モニタリング API を使用して、関連するデータを別個のブラウザから問い合わせるのがベストです。 これは既知の不具合であり、将来のバージョンの Payara Server 5 で修正する予定です。
PrimeFaces のアプリケーションで意図しないエラーが発生する	HTTP/2 プロトコルのサーバー・プッシュ機能は、PrimeFaces のプッシュ・メカニズムに干渉することが判明しています。	すべての関連する HTTP ネットワーク・リスナーでサーバー・プッシュを無効化してください。 現時点では両者を共存させる解決方法ははっきりしていません。そのため、もし他に HTTP/2 のサーバー・プッシュを使用するアプリケーションが存在する場合には、アプリケーションがデプロイされる仮想サーバーをカスタマイズして、個別のネットワーク・リスナーを使用するようにしてください。
HTTP/2 使用時にクライアント証明書による認証が動作しない	設計上、プロトコルは `CLIENT-CERT` 認証をサポートしません。	すべての関連するネットワーク・リスナーで HTTP/2 を無効化してください。 もし HTTP/2 の使用が必須であれば、より適した認証方法に切り替えてください。

まとめ

このガイドで詳細に解説した手順に従えば、古い Payara Server 4 ドメインを Payara Server 5 で動作させることが可能になるはずです。これらの手順は既存のドメインを Payara Server 5 で動作させるために必要な手順を示しているため、必要に応じてアプリケーションの生産性向上のための追加機能についても考慮すべき点に注意してください。これらの機能についての理解を深めるためには公式の製品ドキュメントを参照することをお勧めします。

最後に、このガイドはサーバー移行を実施するに当たってごく一般的なシナリオと課題についてカバーしています。そのため、場合によってはこのドキュメントで触れているものとは異なる問題が発生することもあります。もし商用サポート契約をお持ちであれば、発生した問題に関連する情報をすべて含めてサポート・チケットを上げてください (サポート・ポータルまたはメール support@payara.fish 宛)。課題を解決して移行が成功するまでサポートさせていただきます。

移行に関するさらなるヘルプについて

Payara Enterprise または Migration & Project Support ご契約者様向けのハンズオン・アシスタンス

商用サポートご契約者様には、Payara Server 4 から Payara Server 5 への移行に関して、別途 Payara Accelerator アップグレード・ガイドをご用意しております。こちらの資料では、移行に関するアドオンのコンサルティングについてご説明しております。アップグレード・ガイドをダウンロードしてコンサルティング内容をご確認ください。

Payara Platform 商用サポートのご契約

まだサポート契約をお持ちでなければ、Migration & Product サポートをご契約いただくことで、リスクとコストを抑えつつ、プロジェクトを予定通りかつ予算内に完了できるよう、Payara Platform の専任技術者がお手伝いいたします。

Migration & Project サポートは 1 年の定額契約で、規模に関係なく台数無制限の開発環境をサポートします。お問い合わせに利用できるチケットの数は無制限で、お客様専用のナレッジベースや専用サイトから重要な修正版・パッチをご提供いたします。もし 12 ヶ月以内に商用運用を開

始される場合には、Payara Enterprise 商用サポート・サービスのお値引きもさせていただきます。こちらをクリックして、Migration & Project サポートに関する詳細をご確認ください。

すべての商用サポート・ソリューションに関するカタログはこちらからダウンロードできます。

sales@payara.fish

+44 207 754 0481

www.payara.fish