

Magic xpi 4.x アーキテクチャ



OUTPERFORM THE FUTURE™

はじめに

この文書は、Magic xpi 4.14 の新しいアーキテクチャについて説明しています。Magic xpi 3.x は、Magic リクエストブローカというミドルウェアに基づいていましたが、Magic xpi 4.14 はその基礎をなすメッセージングとコンテキスト持続レイヤとしてインメモリ・データグリッドを使用しています。この新しいアーキテクチャはビルトインのクラスタリングとフェールオーバー機能を Magic xpi プロジェクトに提供し、無制限のリニアスケールビリティを可能にして、パフォーマンスを向上させます。

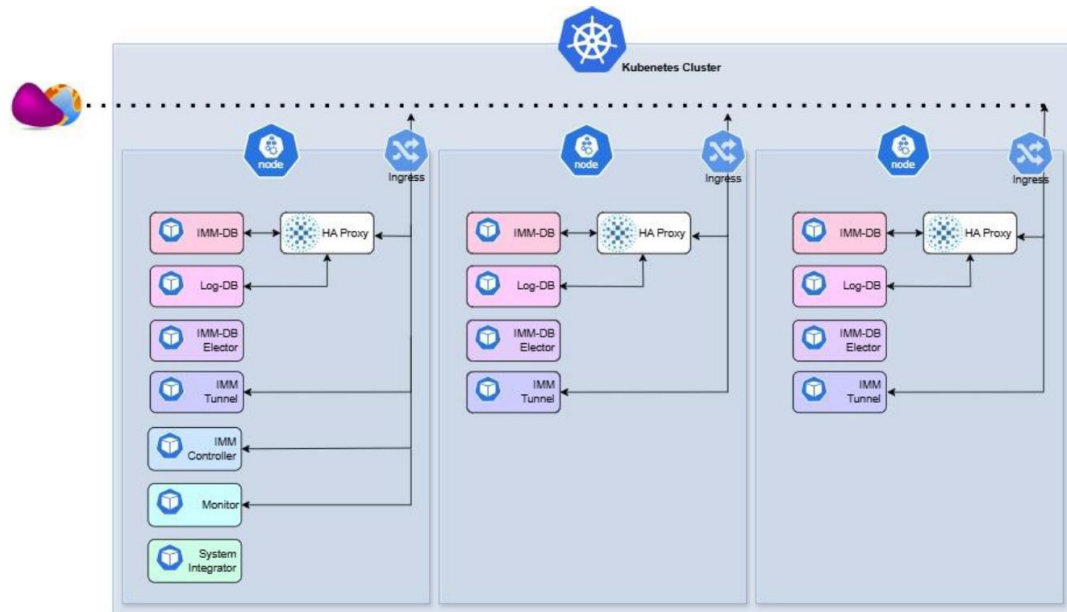
Architecture Overview

Magic xpi サーバは、xpi インストーラで提供される In-Memory Middleware 上に構築されており、このミドルウェアはインフラストラクチャの基礎となるメッセージング層として機能します。xpi サーバは、IMM-Agent を使用して IMM と通信します。1 つの集中型 IMM クラスタと、複数の xpi サーバがこの IMM に接続されています。

以下は IMM の内部レイアウトのイメージです。

IMM 高可用性アーキテクチャ

インメモリミドルウェアは、マスターノードが動作不能になった場合に、ワーカーが引き継いでシステムを中断することなく稼働させ続ける高可用性 (HA) 構成で動作するように設計されています。HAにより、システムのダウンタイムがほぼゼロになり、システムが常に利用可能になります。



The Magic xpi IMM Cluster Layout

The above layout shows the replication and distribution details of all Magic-xpi-Names containers. This will/does not show any of the system required containers as they vary platform to platform.

上の図では、3 つのノードのクラスターが示されています。最上部はマスターで、残りの 2 つはワーカーです。

IMM 用語の理解

IMM は Kubernetes デプロイメントフレームワークに基づいており、複数のサービスが IMM デプロイメント内のノードとしてデプロイされます。

IMM-Agent: xpi プロジェクトサーバ(xpi サーバとも呼ばれます)を実行している各ホストでは、IMM-Agent が実行されている必要があります。このエージェントは、xpi サーバの開始および停止するリクエストを処理し、それらサーバのヘルスチェックを実行します。複数のホストからの IMM-Agent は統合され、IMM を経由して接続されます。

In-memory Middleware を構成するコンポーネントは以下の通りです。:

- **IMM-Controller**
The IMM-Controller は、IMM-Agent が通信し、エンジンの開始および停止の要求を受け入れるプロセスです。
- **IMM-DB**
IMM-DB は、IMM が機能するために必要なすべての運用データが保存されるセントラルリポジトリです。これには、すべての xpi 関連のランタイム データ、プロジェクト メタデータ、およびリアルタイム実行データが含まれます。
- **LOGDB**
LOGDB は、xpi サーバによって生成されるすべてのアクティビティ ログのコンテナとして機能します。

IMM-Tunnel

IMM アーキテクチャ内に存在する Web サービスです。すべての外部リクエスト (HTTP リクエスト、TCP リクエストなど) を処理し、それらを IMM コントローラにリダイレクトします。
注: HTTP Web サーバトリガーなどに必要な Web サーバを置き換えるものではありません。

Monitor

Monitor は、IMM とその下で実行されている xpi プロジェクトに関する情報とヘルス関連データを提供するセントラルインターフェースとして機能します。これにはライセンスの使用状況、サーバの負荷、ワーカなどのグラフなどのデータも含まれます。注: 以前のバージョンの xpi とは異なり、xpi モニタは IMM インフラストラクチャ内に常駐しており、xpi サーバ/スタジオのインストール時に提供されるサービスではありません。

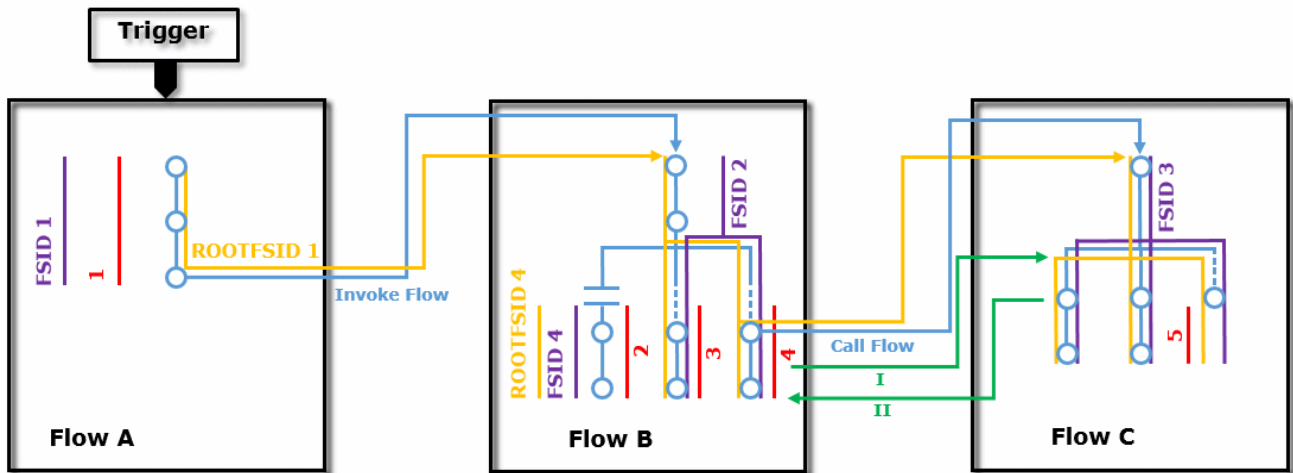
Magic xpi Server

Magic xpi サーバは、Magic xpi 4.x 統合プロジェクトのロジックを実行する複数のサーバプロセスから構成されているアプリケーションサーバソフトウェアです。各 Magic

xpi サーバのプロセス(エンジン)は、複数のスレッド(ワーカ)から構成されています。各ワーカは、さまざまな統合プロジェクト・ロジックを実行することができます。

フロー実行の用語の理解

サーバアーキテクチャについて理解すると、プロジェクトのリカバリ設定を定義する際に役立ちます。以下の図は、サーバアーキテクチャがどのように動作するかを示しています。



ROOTFSID 1

ルートフローシーケンス ID(ルート FSID)は、実行ツリー内のすべてのフローと分岐に対して共通です。

ROOTFSID 4

新しい実行ツリーを開始するスタンドアロンの分岐。

ROOTFSID #

番号は、初期の FSID と同じ数値です。

FSID

各新規フローは、新規フローシーケンス ID(FSID)を受け取ります。スタンドアロンの分岐は、個別のフローとして考えられるので、独自の新しい FSID を取得します。

1...5 = Flow Request ID

各トリガーフロー、並列分岐、またはスタンドアロンの分岐は、独自の ID を持つフローリクエストメッセージによって呼び出されます。「Invoke flow」ステップと「Call flow」の送り先は、リニア実行の一部であり、独自のフローリクエストメッセージを持っていません。

I, II

各 Call flow の繰返しのために、起動されたフロー(I)には新しい FSID が割り当てられます。起動されたフロー(I)が起動元に返ると、フロー呼出(II)は、そのオリジナルの FSID を残します。



Engine Management

Magic xpi4.x サーバのプロセス(エンジン)は、IMM を使用して完全に管理されます。フロー呼出しリクエストを待つフローと同じような方法で各エンジンは IMM 上で管理指示を待ちます。各エンジンにおいて、専門の管理スレッドが、管理メッセージを消費し、処理します。エンジン自身は、リカバリとメンテナンスの目的のために IMM にその動作状態を更新する必要があります。

管理メッセージは、IMM を通してエンジンを管理することができます。エンジンが処理することができるメッセージの例には以下のものが含まれています。:

- ワークスレッドの追加
- ワークおよびスレッドの終了
- トリガの再起動
- シャットダウン
- 実行中のワークへのステータスの提供

スレッドとワーク

Magic xpi 4.x は 2 種類のスレッドを持っています。:

フロースレッド(フローワーク)

これらは、「フローワーク」とも呼ばれます。すなわち、フローを実行することができるスレッドです。フローワークは、フローステップを実行します。フローワークが開始され、IMM のステータスが **READY_FOR_USE** であるフロー呼出しリクエスト(メッセージ)を待ちます。フロー呼出しリクエスト(メッセージ)が作成されたら、有効なフローワークは IMM のステータスを **IN_PROCESS** に変更します。2 つのワークが同じメッセージを実行することを防止するため、それらの実行はトランザクション内で行われ、メッセージペイロード内で有効なデータを使用して、メッセージ内で定義された要求されたフローを実行します。

詳細なモニタリング、トラブルシューティング、そしてメンテナンスを可能にするために、フローワークはその処理の間、IMM に詳細なステータス情報を維持しています。フロー内で各ステップを実行する前に、ワークは現在のフローステータスによって IMM を更新し、プロジェクトメタデータに基づいて、やるべきことをチェックします。:

- ユーザ定義タイムアウトに達したらアボートする
- リカバリまたはエラー処理のためのアボート
- サーバ/プロジェクトシャットダウンのためのアボート
- (デバッグ状態)で次のステップを続ける前にポーズ

フローが完了したら、ワークは IMM のフロー呼出しリクエストメッセージのステータスを **DONE** に更新し、追加の新しいメッセージが処理されるのを待ち始めます。

新しいプロセスが開始される前に、ビジネスプロセスが完了することを保証して IMM からのメッセージを読み込む時、**Magic.ini** ファイルの **WorkerReadDeeperMessagesfirst= flag** を設定することで、Magic xpi に、より高い階層を持つメッセージに優先度を付けるように指示します。

i ルートメッセージのための処理は、先入れ先出し(FIFO)ですが、実行ツリー(並行ステップ)により深いメッセージはより高い優先事項を持っていて、待機中のルートメッセージよりも前に処理されます。この仕組みにより、新しいルートメッセージが処理される前に、実行ツリーをより迅速な完了することができます。

トリガスレッド (トリガーワーカ)

これらは、外部イベントをポーリングまたは「リスニング」し、IMM でフロー呼び出し要求を作成してフローを開始する責任を持つスレッドです。トリガスレッドが開始され、外部のイベントが発行されるのを待ちます。そのようなイベントが発生すると、トリガはイベント データ (ペイロード) を含むフロー呼び出しリクエスト (メッセージ) を作成し、利用可能なフロー ワーカによって処理されます。トリガは同期または非同期にすることができます。:

- **同期型トリガ** - 一部のトリガはフロー リクエストを非同期的に呼び出しますが、他のトリガには同期動作モードがあります。このモードでは、トリガがフロー呼び出し要求を作成した後、応答メッセージを待ちます。同期リクエストを処理するフロー ワーカは、フローの完了後にペイロードを含む応答メッセージを作成する責任があります。
- **非同期型トリガ** - これらのトリガはフロー リクエスト メッセージを作成し、応答を待たずに外部システム内の新しいイベントのチェックを直ちに続行します。非同期トリガが未処理のフロー要求メッセージで IMM があふれさせるのを防ぐために、各トリガには事前定義されたキュー バッファサイズがあり、キューが事前定義されたサイズを超えた場合に新しいイベントの作成を抑制するために使用されます。このトリガバッファ サイズのデフォルト値は 10 です。この値は、`TriggersBufferSize`= フラグの値を変更することで調整できます。フロー ワーカと同様に、トリガは IMM 内で状態を維持し、保守性と可視性を確保します。

i 各 Magic xpi ワーカは、特定の Magic xpi プロジェクトのリクエストの処理専用です。同じ IMM 上で複数のプロジェクトを実行する場合、各プロジェクトに専用の Magic xpi エンジンとワーカを割り当てる必要があります。

外部トリガ

外部トリガは、マネージド エンジンのトリガー ワーカー スレッドでは実行されません。代わりに、それらは独自のプロセスで実行され、他のアプリケーションからのフロー呼び出しリクエストを作成します。

たとえば、HTTP トリガは、Web サーバ (IIS または Apache) で実行されるモジュールです。SOAP Web サービス トリガは、SOAP サーバによって呼び出される外部トリガーのもう 1 つの例です。外部トリガーは通常は同期型です。

実行コンテキスト

完全なプロジェクトのスケールビリティと場所の独立性を可能にするために、Magic xpi 4.x はプロジェクトのメタデータとランタイム コンテキストの一部を、Magic xpi 3.x のようにローカル プロセス メモリではなく IMM にロードします。

これにより、グリッドに参加しているすべてのマシン上で、プロジェクトを実行しているすべての Magic xpi サーバ/ワーカがすべてのコンテキスト データを利用できるようになります。これにより、特別な設計を考慮することなく、任意のプロジェクトを任意の数のマシンに自動的に拡張できます。

スタートアップ

Magic xpi 4.14 のスタートアップは下記の要素を含んでいます:

Magic xpi プロジェクトを開始/停止する

1. プロジェクトの開始は、以下の方法のうちの 1 つで手動で行うことができます:
 -  **Start** リンクをクリックします。プロジェクトがビルドされると、このスタートリンクはプロジェクトのディレクトリに作成されます。リンクは、start.xml 設定ファイルを示しています
 - モニタまたはデバッガから**スタート**オプションをクリックします。このオプションも、プロジェクト・フォルダ内の **start.xml** ファイルを使用します。
2. ステップ 1 のオプションは、プロジェクトの Start.xml ファイルのメタデータ、一意の ID、および **START_REQUESTED** のステータスを使用して、IMM にサーバエンティティ(または複数のエンティティ)を作成します。(サーバエンティティとステータスはモニタで確認できます。)プロジェクトを開始する度に、エンティティの別にインスタンスが IMM に作成されます
3. IMM コントローラは、IMM をスキャンして、ステータスが **WAITING_FOR_AGENT** であるサーバエンティティを探します
4. IMM コントローラは、**WAITING_FOR_AGENT** のステータスを持つサーバエンティティを見つけると、Start.xml ファイルに基づいて、サーバエンティティで定義されたホスト名または IP アドレスに従って IMM をスキャンします。コントローラは、このサーバエンティティで定義されている全てのパラメータ/起動情報を IMM エージェントに渡します。
5. サーバエンティティで指定されているホスト・マシンで実行する IMM-Agent は、IMM-Controller コントローラから渡されるパラメータによって、Magic xpi サーバ (**MgxpiServer.exe** プロセス、別名エンジン)を起動します。
 - プロジェクトが実行していない場合、開始する最初の **MgxpiServer.exe** プロセスは IMM にプロジェクト・メタデータを読み込ませることによってプロジェクトを作成します。

- プロジェクトがすでに実行している場合、読み込まれた **MgxpiServer.exe** プロセスは既存のプロジェクトを追加し、自身のワーカとトリガを要員として追加します。
6. Magic xpi サーバに渡されるパラメータの一つは、IMM に作成されたサーバエントリの ID です。Magic xpi サーバが登録される時に IMM に接続され、指定された ID のサーバエントリを探し、ステータスを **RUNNING** に更新します。
 7. サーバはリクエストの処理を開始します。

プロジェクトの停止は、以下に行います。:

-  **Stop** リンクをクリックします。プロジェクトがビルドされると、この**停止**リンクはプロジェクトのディレクトリに作成されます
- モニタまたはデバッガから**停止**オプションをクリックすることによって実行されます。

スタートアップ失敗

サーバエンティティのステータスが、**START_REQUESTED** として残っていて、**RUNNING** に変わらない場合、Magic xpa プロセスユニットは IMM-Agent とチェックして、実際のプロセスが動作しているかどうかを確認します。プロセスが動作している場合、サーバエンティティのステータスは **START_FAILED** に変わります。プロセスが動作している場合、Magic xpa プロセスユニットは、最初に IMM-Agent にプロセスを終了させるように指示します、そして、サーバエンティティを **START_FAILED** に更新します。

プロジェクトの実行シーケンス

以下の手順は、プロジェクトのステータスが **RUNNING** の場合に発生するイベントの手続きを説明しています。:

1. プロジェクトが実行しているとき、3 つの要素が同時に動作します: ワーカ、ポーリングトリガと外部トリガです。各 Magic xpi サーバは、いろいろなタスクのために一つ以上のワーカ/トリガを実行することができます:
 - a. ポーリング・トリガ (Magic xpi サーバスレッド) は、フローを呼び出す必要があるかどうかを確認するために、絶えず外部システム (例えば、e メールアカウント) をスキャンします。フローを呼び出す必要がある場合、IMM に **READY_FOR_USE** のステータスのフローメッセージを置きます。
 - b. 有効な各ワーカはジョブの実行に対して IMM を検索します。これは、ステータスが **READY_FOR_USE** であるメッセージを意味します。ワーカがメッセージを見つけると、ステータスを **IN PROCESS** に変更し、フローにメッセージ・ペイロードを渡して、フローロジックを実行します (これは、プルメカニズムとして知られています)

i 2 つのワーカが同じメッセージを実行することを防止するために、1 つのワーカだけが成功するように、**IN PROCESS** のステータスの変更はトランザクション内で実行されます

- c. 外部トリガは、外部アプリケーション（例：HTTP リクエストや Web サーバ）です。一旦トリガがリクエストを受けると、それはフローを呼び出す IMM にメッセージを置きます。同期外部トリガも応答メッセージを待ちます。
2. メイン フローが完了すると、ワーカは IMM 内のリクエスト メッセージのステータスを DONE に更新し、追加のメッセージを自由にスキャンできるようになります。
3. メッセージが同期トリガ（HTTP トリガなど）からのものである場合、フローが終了すると、ワーカは応答メッセージを IMM に書き込み、トリガはそれをクライアントに送信します。
4. すべての並列およびスタンドアロン ブランチも、フローによって IMM に書き込まれる別個のメッセージとして処理されます。この新しいスレッド（並列またはスタンドアロン ブランチ）は、プロジェクトのどのワーカでも（別のマシンで実行されているワーカでも）処理できます。

Magic xpi プロジェクトのシャットダウンシーケンス

適切なタイムアウト値を指定することで、プロジェクトを正常にシャットダウンできます。シャットダウン コマンドが送信されると、タイムアウトに達するまで、プロジェクトは新しいリクエストを処理しません（すべてのトリガが停止します）。ただし、プロジェクトは、すでに受け取った作業の処理は続行します。

バッチから複数のプロジェクトを起動する

プロジェクトの Start リンクを調べると、以下のようなコマンドが表示されます。:

```
"<Magic xpi installation>%Runtime%MgxpiCmdl.bat" start-servers -startup-config-file "<Magic xpi installation>%Runtime%projects%Project1%start.xml"
```

このコマンドは、プロジェクトの下にある特定の start.xml ファイルを指しています。同じコマンドを変更して、複数のプロジェクトをロードできる構成の別の start.xml ファイルをロードすることができます。

リカバリ動作

Magic xpi 4.x は、自動的に多くの障害と故障のシナリオから復旧することができます。

エンジン/ワーカ障害

リカバリのメカニズムには下記の 3 つの要素があります。:

識別

各エンジンとそのワーカは、そのステータスを IMM に報告します。IMM はそれらの報告を監視し、設定されたタイムアウト内にステータスを報告しなかったワーカやエンジンを見つけた場合、次の様に動作します。:

- フローワーカ(トリガまたはフロー)の場合、それはワーカのステータスのために実行しているエンジンにワーカを問い合わせます。
- エンジンの場合、エンジンがまだ動作している場合、このエンジンを起動した IMM-Agent に問い合わせます。

エンジンとワーカのリカバリ

応答がないか存在しないエンジンが発生した場合、IMM エージェントは既存のエンジンを終了して、新しいものを開始するように指示します。スレッドがクラッシュした場合、エンジンは同じエンジンの配下で新しいワーカを開始するように指示します。

ワークプロセスのリカバリ

ワークプロセスは、ルートフロー全体(その子フローを含みます)の実行ツリーです。ワークプロセスは、全体として 1 つの企業取引として扱われます。ルートフローは、親フローによって呼び出されなかったフローです。フローはトリガ、スケジューラ、自動起動、またはパブリッシュ/サブスクライブのシナリオによって開始されたものになります。

1. リカバリは、個別のフローのためにではなくワークプロセスのために定義されます。
2. Magic xpi は、ルートフローで定義されたリカバリポリシーを使用して、子フローのために定義されたリカバリポリシーを無視します。

i スタンドアロンの分岐はワークプロセスから分離されるため、ワークプロセスの一部とみなされません。従って、それは独自の実行ツリーを持っています。スタンドアロンの分岐がクラッシュすると、リカバリポリシーは主要なフローからではなく自身のフロープロパティから取り出されます。

3. セーブポイントは、最上位のフローのリニアな分岐に保存されるだけです。子フローまたは子コンテキストに定義されたセーブポイントは無視されます。
4. ワークプロセスは、(並行分岐の場合は)一つ以上のワーカによって実行することができ、一つ以上の個別の物理サーバで実行します。

5. 現在実行しているワークプロセスの一部がクラッシュすると、ワークプロセスの全体がアボートされます。そして、最上位のフローで定義されるクリーンアップ・フローが呼び出されます。
6. リカバリが適用される場合:
 - a. アボートの場合 …… 更なるアクションは不要です
 - b. 再起動の場合 …… Magic xpi は、最初に起動されたペイロードと一緒に最上位のフローを再開します。最初のデータは、決して失われません。
 - c. セーブポイントの場合 …… 最後のセーブポイントが存在しない場合、Magic xpi は最後のセーブポイントから起動するか、フローを再起動します。

トリガーリカバリー

Magic xpi の各トリガには、オプションでキープアライブのプロパティがあります。トリガが障害とみなされ、再起動する必要があるまで、このプロパティは待ち時間をコントロールします。

2つの例外は、外部トリガである Web Service トリガと HTTP トリガです。これらのトリガは外部プロセス (IIS/Apache/Systinet サーバ) で動作して、モニタすることができて、個別に管理することができます。

About Magic Software Enterprises

Magic Software Enterprises (NASDAQ: MGIC) empowers customers and partners around the globe with smarter technology that provides a multi-channel user experience of enterprise logic and data.

We draw on 30 years of experience, millions of installations worldwide, and strategic alliances with global IT leaders, including IBM, Microsoft, Oracle, Salesforce.com, and SAP, to enable our customers to seamlessly adopt new technologies and maximize business opportunities.

For more information, visit www.magicsoftware.com.



Magic is a registered trademark of Magic Software Enterprises Ltd. All other product and company names mentioned herein are for identification purposes only and are the property of, and might be trademarks of, their respective owners. Magic Software Enterprises has made every effort to ensure that the information contained in this document is accurate; however, there are no representations or warranties regarding this information, including warranties of merchantability or fitness for a particular purpose. Magic Software Enterprises assumes no responsibility for errors or omissions that may occur in this document. The information in this document is subject to change without prior notice and does not represent a commitment by Magic Software Enterprises or its representatives.

© Magic Software Enterprises, 2023 – 2025

