Magic xpi 4.9
アーキテクチャー

OUTPERFORM THE FUTURE™

はじめに

このドキュメントでは Magic xpi 4.9 の新しいアーキテクチャーをについて説明します。Magic xpi 3.x までは ミドルウェアとして Magic Requester Broker を使用していましたが Magic xpi 4.9 では基盤となるメッセージとコンテキストを維持する層として、インメモリー データ グリッドを採用しています。この新しいアーキテクチャーにより、Magic xpi プロジェクト実行時におけるクラスタリングとフェイル オーバー機能を提供します。これにより無制限のスケーラビリティの向上とパフォーマンスの向上を実現します。

アーキテクチャー概要

Magic xpi 4.9 は以下のソフトウェアコンポーネント/要素で構成されています。:

インメモリー データ グリッド (IMDG)

インメモリ データ グリッドは、複数のマシン インスタンス（物理または仮想）上で動作する複数のサーバー・プロセスで構成されたミドルウェア ソフトウェアであり、大量のデータをメモリーに格納し、高性能、弾性スケーラビリティ、フェールセーフ冗長性を実現します。

Space(スペース)

スペースは、データグリッド内で実行されるデータおよびビジネスロジックコンテナ（データベースインスタンスに似ています）です。データグリッドには複数の Spaces を含めることができます。 Magic xpi 4.9 では、複数のプロジェクトを実行するために、2 つのメインスペースと、データベースにミラーリングするための 3 つ目のスペースが使用されます。冗長性とスケーラビリティのために、Spaces のデータとビジネスロジックは、データグリッドに参加しているすべてのマシンに複製され、分割され、マシンまたはソフトウェアに障害が発生した場合でも継続的なサービスを保証します。

Magic xpi サーバ

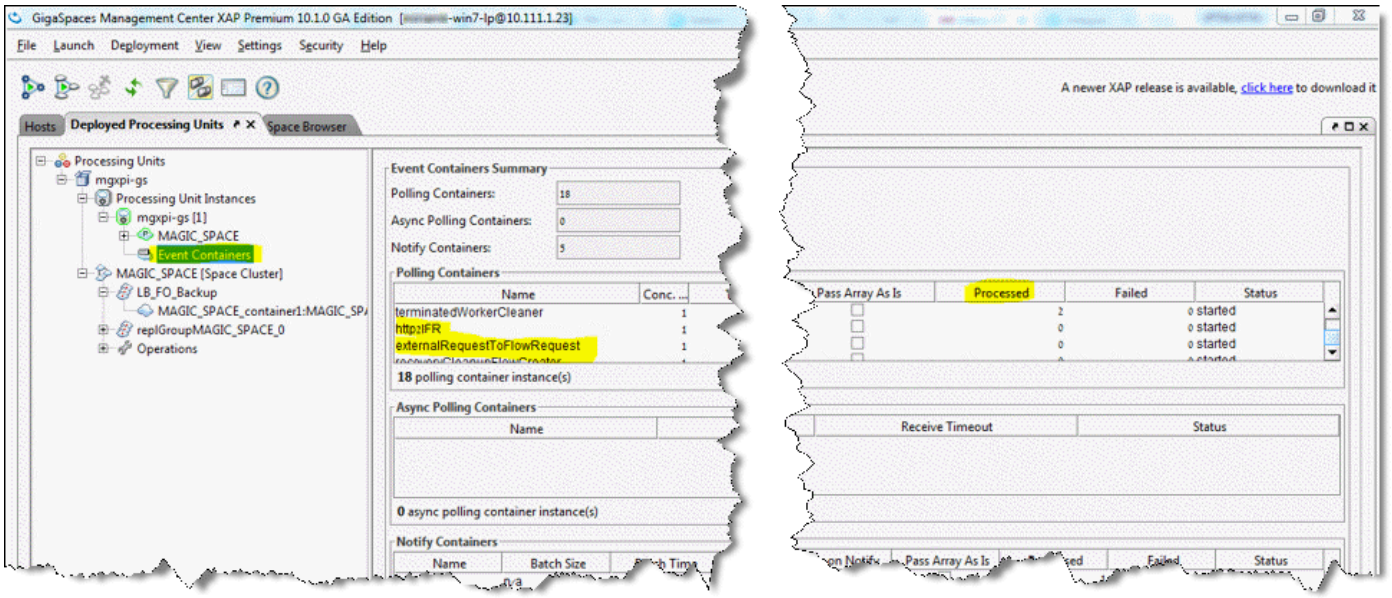
Magic xpi サーバは、Magic xpi 4.9 インテグレーション プロジェクト ロジックを実行する複数のサーバ プロセスで構成されたアプリケーションサーバソフトウェアです。 各 Magic xpi サーバ プロセス(エンジン)は、複数のスレッド(ワーカー)で構成されています。 各ワーカーは、任意のインテグレーション プロジェクト ロジックを実行することができます。

Magic Processing Unit (PU)/Magic 処理ユニット(PU)

Magic PU は、Space で実行され、プロジェクトオブジェクトに対してさまざまな管理タスクを実行するソフトウェアモジュールです。この PU は全ての Magic xpi オブジェクトを監視/管理し、サーバが正常に動作していることを確認します。Magic PU には以下の機能があります:

- フローがタイムアウトしたことを認識し、リカバリーする
- ワーカーおよびサーバがハング/クラッシュしたことを認識し、リカバリーする
- 実行中のサーバに対し、管理メッセージを発行する
- 完了したフロー リクエストのクリア
- プロジェクト エンティティの統計情報の収集

さまざまな PU が GigaSpaces UI の **Event Containers** に表示されます。有用な追加情報が GigaSpaces UI には表示されます。たとえば、**Processed** 列には、次のスクリーンショットに示すように、各 PU で処理された要求の数と発生したタイムアウトの数などが表示されます。

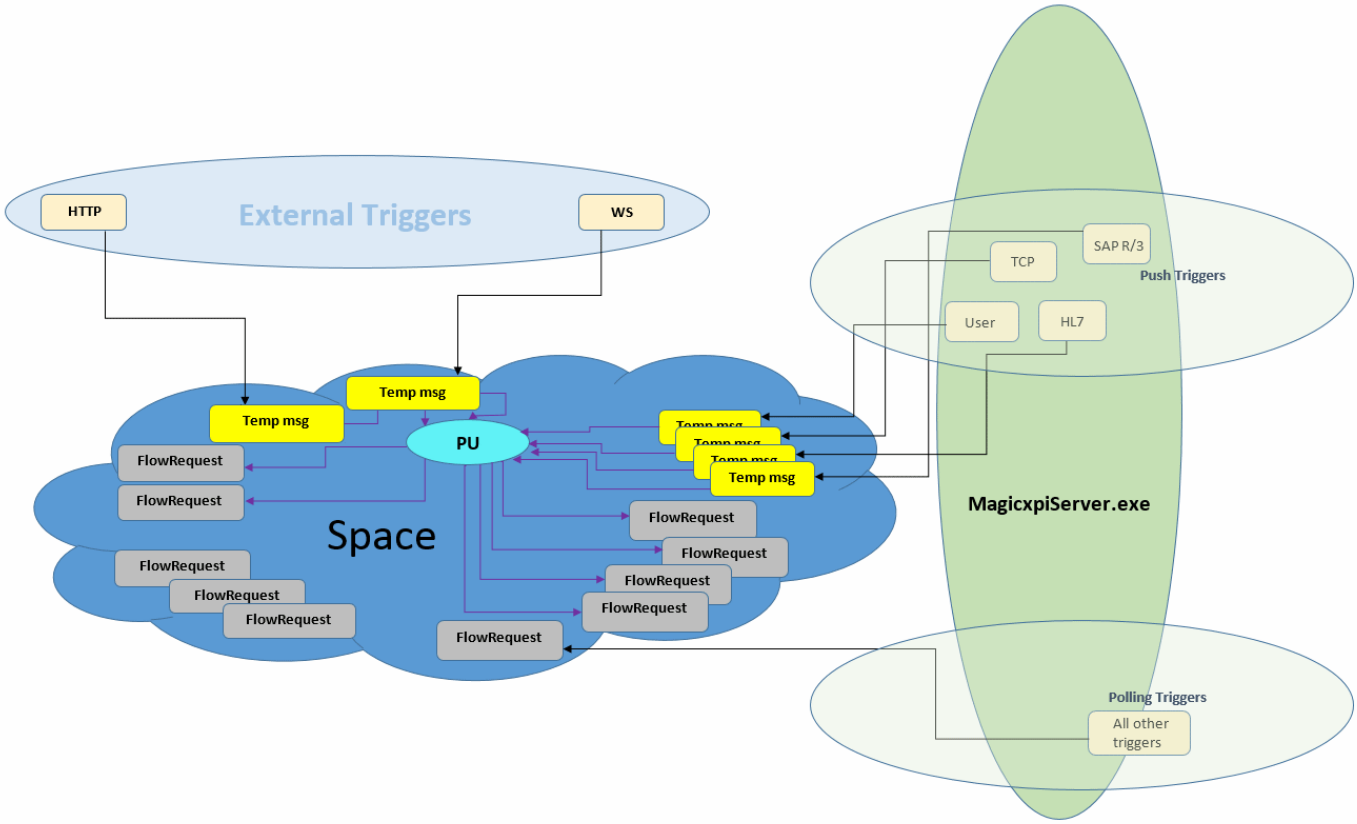


メッセージ フロー

Magic xpi エンジンはスペースに接続するクライアント アプリケーション ソフトウェア モジュールである GigaSapces プロキシを通じてスペースと通信します。

各 Magic xpi エンジンはフロー スレッドとトリガー スレッドを実行することができます。さらに、HTTP トリガーや Web サービス トリガーなど、他のプロセス スペースで実行されている外部トリガーもあります。

トリガーは新たにフロー起動リクエスト メッセージを生成し、そのメッセージは同期あるいは非同期の処理としてワーカーに処理されます。次の図は Magic xpi のトリガー アーキテクチャーを示しています。



外部トリガー

Magic xpi はさまざまなリクエストを受け取ることができます。:

- HTTP IIS に到着する Magic xpi へのリクエストは、<Magic xpi 4.9>\Runtime\scripts\config\mgreq.ini ファイルに従って、<Magic xpi 4.9>\Runtime\scripts\bin\MgWebRequester.dll ファイルによって処理されます。
- Apache Tomcat サーバ(Java)に到着する Magic xpi へのリクエストは、-Dcom.magicsoftware.requester.conf パラメータの

CATALINA_HOME\bin\startup.bat file ファイルで定義されている mgreq.ini ファイルに従って、<Magic xpi 4.9>\Runtime\Support\JavaWebRequester\TomCat\magicxpi4.war ファイルによって処理されます。(Magic xpi - Java-Based Installation Instructions.pdf をご覧ください)。

- Systinet サーバ (WS サーバ)に到着する Magic xpi へのリクエストは、-Dcom.magicsoftware.requester.conf パラメータの<SSJ for Magic xpi>\bin\server.bat ファイルに定義されている mgreq.ini ファイルに従って、<SSJ for Magic xpi>\lib\MgRequester.jar ファイルによって処理されます。

  これらの全ては、Temp msg (上図で黄色で示されている)をスペースに投入します。

  注: 各トリガータイプには固有の Temp msg タイプがあり、それらは GigaSpaces UI で見ることができます。上図において、アーキテクチャーを説明するために全トリガーに Temp msg が使用されています。

プッシュ トリガー

プッシュトリガーから到着した Magic xpi のリクエストは、Magic xpi サーバプロセス内の(Mgrequest クラスの)<Magic xpi 4.9>\Runtime\java\lib\uniRequester.jar ファイル(mgreq.ini ファイルは関係ありません)によって処理され、Temp msg がスペースに挿入されます。これは、別プロセスでリクエストを処理する WS/HTTP とは異なります。

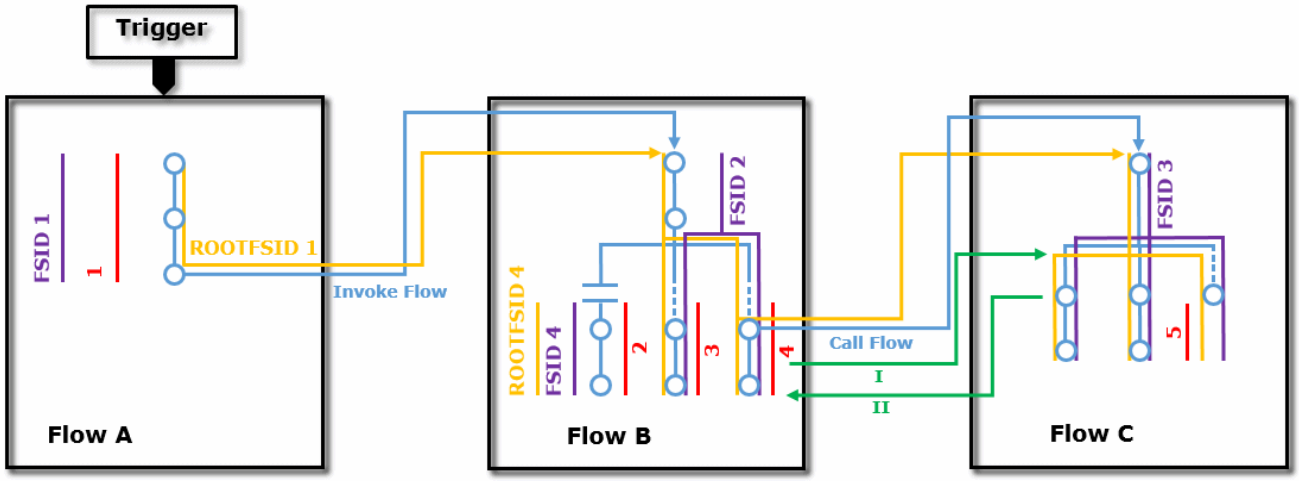
ポーリング トリガー

トリガーを起動する他の全てのリクエストは、FlowRequest メッセージを Space に直接書き込む Magic xpi サーバプロセス(mgreq.ini ファイルは関係ありません)内で処理されます。MagicxpiServer.exe ファイルの利用可能なワーカーは、プロジェクトの制約(最大インスタンス、ライセンスなど)を考慮して、FlowRequest メッセージを取り出して実行します。
*ユーザ リクエスト = mgrqcmdl, CallRemote

処理ユニット (PU)

スペースにはこれらの Temp msgs を FlowRequest メッセージに変換する 2 つの専用 PU(http2ifr という名称の HTTP トリガータイプ用の PU と、それ以外で使用される externalRequestToFlowRequest* という名称の PU)が存在します。これらの FlowRequest メッセージは MagicxpiServer.exe ファイルによって処理され、Monitor および GigaSpaces UI(データタイプ名は com.magicsoftware.xpi.server.messages.FlowRequest)に表示されます。
* http2ifr と externalRequestToFlowRequest PU は GigaSpaces UI の Event Containers セクションに表示されます。

サーバのアーキテクチャーを完全に理解することは、プロジェクトのリカバリー設定を行う際に非常に有用です。下図はサーバ アーキテクチャーの仕組みを示しています。



ROOTFSID 1	ルートフローシーケンス ID(ルート FSID)は、同じランタイムツリー内では全てのフローと分岐で同じ値です。
ROOTFSID 4	スタンドアロンの分岐は新たなランタイムツリーを開始します。
ROOTFSID #	最初の FSID と同じ番号です。
FSID	新しい各フローは、新しいフローシーケンス ID(FSID)を受信します。スタンドアロンの分岐は別のフローと見なされるため、別の FSID が振られます。
1...5 = Flow Request ID	トリガーされた各フロー、パラレルの分岐やスタンドアロンの分岐は個別の ID を持つ FlowRequest メッセージから起動されます。"フロー呼出(Invoke flow)"ステップやデータマッパ送り先での"call flow"はリニア実行の一部なので、独自の FlowRequest メッセージを持つわけではありません。
I, II	call flow の度に、called flow (I)に新しい FSID が割り当てられます。called flow が calling flow (II)に戻る時、calling flow (II)は元の FSID を保持します。

エンジン管理

Magic xpi 4.9 サーバ プロセス(エンジン)は、スペースを使用して完全に管理されます。各エンジンは、フロー起動リクエストを待っているフローと同様に、管理命令のためスペースを待機しています。各エンジンでは専用の管理スレッドが管理メッセージを処理します。また、エンジン自体は、リカバリーおよびメンテナンスの目的で、スペースへの実行ステータスの更新を行います。

管理メッセージを使用すると、スペースを介してエンジンを管理できます。エンジンが処理できるメッセージの例を次に示します。:

- ワーカースレッドの追加
- ワーカーの停止
- トリガーの再起動
- シャットダウン
- 実行中のワーカーのステータスの提供

スレッドとワーカー

Magic xpi 4.9 には 2 つの異なるタイプのスレッドが存在します:

フロー スレッド (フロー ワーカー)

これらは"フロー ワーカー"とも呼ばれ、フローを実行可能なスレッドです。フロー ワーカーはフロー ステップを実行します。フロー ワーカーは、スペース内のステータスが READY_FOR_USE であるフロー起動リクエスト(メッセージ)待機し、開始します。フロー起動リクエスト(メッセージ)が作成されると、実行可能なフロー ワーカーは、スペース内のステータスを IN_PROCESS に変更します。2 つのワーカーが同じメッセージを実行するのを防ぐためにトランザクション内でこの処理を行い、メッセージペイロードで利用可能なデータを使用して、メッセージに定義されているリクエストされたフローを実行します。

フロー ワーカーは詳細なモニタリング、トラブルシューティング、メンテナンスを実現するために処理の詳細なステータスを保持しています。フローの各ステップを実行する前に、ワーカーはスペースを現在のフローステータスで更新し、次にプロジェクトメタデータに従って、以下が必要かどうかをチェックします。:

- ユーザが定義したタイムアウトに達したことによるアボート
- リカバリーあるいはエラー ハンドリングによるアボート
- サーバ/プロジェクトの停止によるアボート
- (デバッグ状態で)次ステップに進む前の一時停止時

フローが完了すると、ワーカーはスペース内のフロー起動リクエスト メッセージのステータスを DONE に更新し、新たなメッセージを処理できるよう待機状態になります。

Magic.ini ファイルの WorkerReadDeeperMessagesfirst=フラグは、スペースからのメッセージを読むときに Magic xpi にメッセージの優先度を指定し、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ようにします。

注: ルートメッセージの処理方法は先入れ先出し(FIFO)ですが、実行ツリーの深いメッセージ(パラレルステップ)は優先順位が高く、待機中のルートメッセージの前に処理されます。この方式では、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すばやく完了することができます。

トリガー スレッド (トリガー ワーカー)

これらは、外部イベントへのポーリングまたはリスニングを担当するスレッドであり、スペースでフロー起動リクエストが作成されることによってフローを開始します。 トリガースレッドは、開始すると、外部イベントが発生するのを待機します。 このようなイベントが発生すると、トリガーはイベントデータ(ペイロード)を含むフロー起動リクエスト(メッセージ)を作成し、処理可能なフローワーカーによって処理されます。 トリガーは、同期または非同期にすることができます。:

- **同期トリガー** – トリガーの中には、フローリクエストを非同期に呼び出すトリガーもありますが、トリガーには同期動作モードが準備されています。このモードでは、トリガーは、フロー起動リクエストを作成した後、レスポンス メッセージを待機します。同期リクエストを処理するフロー ワーカーは、フローが完了するとペイロードを含むレスポンス メッセージを作成します。
- **非同期トリガー** – これらのトリガーは、フロー リクエスト メッセージを作成し、レスポンスを待つことなく、すぐに外部システムの新しいイベントをチェックし続けます。未処理のフロー リクエスト メッセージでスペースが溢れ出さないよう、各トリガーにはあらかじめ設定されたキュー バッファー サイズが設定されており、所定の値を超えた場合、新しいイベントをの作成が抑制されます。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 ワーカーと同様にトリガーは保守性と可視性確保のためスペース内にステータスを保持します。

注: 各 Magic xpi ワーカーは特定の Magic xpi プロジェクトのための処理要求専用に使用されます。同一の IMDG/space で複数のプロジェクトが実行する場合、各プロジェクトには専用の Magic xpi エンジンとワーカーを割り当てる必要があります。

外部トリガー

外部トリガーは、管理対象エンジン内のトリガー ワーカー スレッドでは実行されません。 代わりに、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発信によるフロー起動リスエストを作成します。

例えば HTTP トリガーは Web サーバ(IIS や Apache)配下で実行されるモジュールです。SOAP Web サービス トリガーは Systinet サーバにより呼び出される外部トリガーの例です。通常、外部トリガーは同期通信となります。

ランタイム コンテキスト

Magic xpi 3.x でローカル・プロセス・メモリーに読み込んでいたのとは違い、プロジェクトのスケーラビリティと場所の独立性を向上させるため、Magic xpi 4.9 ではプロジェクトのメタデータとランタイム コンテキストをスペースに読み込みます。

このことにより、グリッドを構成する全てのマシン上で、プロジェクトを実行している全ての Magic xpi サーバ/ワーカーが全てのコンテキストデータを利用/共有が可能になります。これにより、特別な設計上の考慮することなく、任意のプロジェクトを複数のマシンに自動的に拡張縮小することができます。

スタートアップ

Magic xpi 4.9 のスタートアップは以下の要素より構成されます。:

Magic xpi GSA サービスの実行

Magic xpi 4.9 をインストールする際、**Magic xpi GSA** (Grid Service Agent) と呼ばれる OS サービスのインストールを強くお勧めします。Windows の場合は OS サービスとして、UNIX の場合は init デーモン(注：UNIX 版は日本では提供されません)として実行されます。

Magic xpi GSA サービスは最初にグリッドをロードし、その後、Magic Space を実装します。これら 2 つのモジュール(グリッドと Magic Space)は、GigaSpaces のインフラストラクチャモジュールです。Magic xpi 4.9 プロジェクトは、これら 2 つのモジュールが実行されていないと実行できません。

GSA サービスの手動インストール

インストール時に GigaSpaces GSA サービスの自動インストールを選択しなかった場合、以下のコマンドで手動でインストール/アンインストールを行うことができます:

- **installService.bat** (GSA のインストール)
- **uninstallService.bat** (GSA のアンインストール)

これらのファイルは以下のパスに存在します:

<Magic xpi 4.9 インストール先>\Runtime\OS_Service\scripts

Windows 7 以降のオペレーティングシステムでは、管理者の資格情報を使用してこれらのコマンドを実行する必要があります。

スタートアップ時に自動実行される Magic xpi GSA サービスの設定

ご利用のコンピューターの起動時に自動的に GSA サービスが開始するよう、OS サービスの設定を行うことをお勧めします。GSA サービスを自動起動するには以下の手順を実行します。:

1. **スタート** ボタンをクリックし、 **ファイル名を指定して実行** をクリックします。 **実行** ダイアログボックスが開きます。
2. **実行**ダイアログボックスで、services.msc と入力します。**サービス**ダイアログボックスが開きます。
3. サービス ダイアログボックスで、Magic xpi 4.9 GSA をダブルクリックします。 Magic xpi 4.9 GSA **プロパティ (ローカルコンピュータ)**ダイアログボックスが開きます。
4. Magic xpi 4.9 GSA **プロパティ (ローカルコンピュータ)** ダイアログボックスの**スタートアップの種類**パラメータで、ドロップダウンリストから**自動**を選択します。
5. OK をクリックし終了します。

コマンドラインから Magic xpi GSA サービスを直接実行する

コマンドラインで GSA サービスを起動し、Space を実装するには:

- magicxpi_gs-agent.bat ファイルを実行して、グリッドをロードします。Magic xpi GSA サービスが自動的に展開するように設定されている場合、自動的に Space も実装されます。そうでない場合は、Magicxpi_deploy.bat ファイルを使用して Magic Space を実装します。

これらのファイルは、次のパスの下にあります。:

<Magic xpi 4.9 インストール先>\Runtime\GigaSpaces\bin

※ Excel/Word アダプタを使用する際は必ず手動で GSA サービスを起動する必要があります。

GigaSpaces グリッドのロード

特定のマシンで Magic xpi GSA サービスが開始されると、グリッドはローカルにロードされ、LAN 内で同じ Lookup Locator 名を持つ他のグリッドコンポーネントが検索されます。そのようなコンポーネントが見つかった場合、ローカルグリッドはそのルックアップロケータの一部とみなされます。このようにして、1 つの統合されたグリッドがネットワーク内に確立されます。

注: Magic xpi エンジンは、ログインしたユーザではなく、GSA サービス用に定義されたユーザのもとで実行されるので、注意が必要です。デフォルトでは、ローカルシステムアカウントでサービスが起動されます。1 台のマシンで Magic xpi を実行するには、通常これで問題ありません。 ただし、クラスタ環境では、ネットワークリソースにアクセスする権限を持つユーザでサービスを実行する必要があります。

グリッドが実行されていることを確認するには:

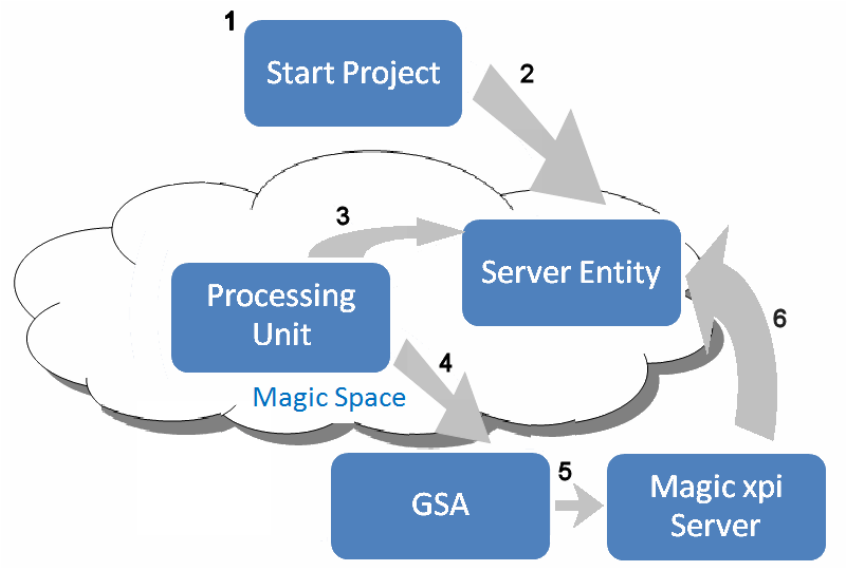
1. インストール ショートカットより GigaSpaces UI を実行します。複数タブで構成される GS ユーティリティ画面が開きます。
2. グリッドが正常に開始されている場合は、マシンが **ホスト** タブに表示されます。 インストール時に **開発マシン** チェックボックス(デフォルトオプション)を選択すると、マシン名の下に GSA が 1 つ、GSC が 2 つ、GSM が 1 つ、LUS が 1 つ必要です。 グリッドが正常に起動しなかった場合は、LookupGroupName=フラグと LookupLocators=フラグが正しく構成されていることを確認する必要があります。 これらのフラグは両方とも Magic.ini ファイルの[MAGICXPI_GS]セクションにあります。

Magic Space の実装

グリッドが起動して実行されると、Magic xpi GSA サービスは自動的にグリッド上の Magic Space を実装します。Magic Space がグリッド上にどのように実装されるか、例えば、パーティションの数、バックアップの数等、デフォルトでは以下のフォルダーに配置されている magicxpi_sla.xml ファイルで定義されています。:

<Magic xpi 4.9 インストール先>\Runtime\config

Magic xpi プロジェクトの開始と停止



1. プロジェクトの開始は、以下のいずれかの方法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。Start リンクはプロジェクトのビルド時にプロジェクト フォルダに作成されます。このリンクは start.xml 設定ファイルを指しています。
   - モニタあるいはデバッガの Start オプションをクリックします。このオプションはプロジェクト フォルダ内の start.xml ファイルでも使用されます。

   あるいは 自動起動:

   - projectsStartup.xml ファイルを作成し、**<Magic xpi インストール先>\Runtime\Config** フォルダに保存します。

     Magic xpi サービスが起動し、Magic Space の実装を管理すると、自動的に projectsStartup.xml ファイルにリストされているプロジェクトとサーバが起動します。projectsStartup.xml の構造は、各プロジェクトで作成された start.xml の構造と同じです。

2. 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と一意の ID と START_REQUESTED の状態で Magic Space にサーバエンティティ(または複数)を作成します。(サーバーエンティティとステータスは Monitor で確認できます。) プロジェクトを開始する度に Magic Space 内に別のエンティティのインスタンスが作成されます。
3. Magic PU は、Magic Space をスキャンして、ステータスが START_REQUESTED のサーバ エンティティを探します。

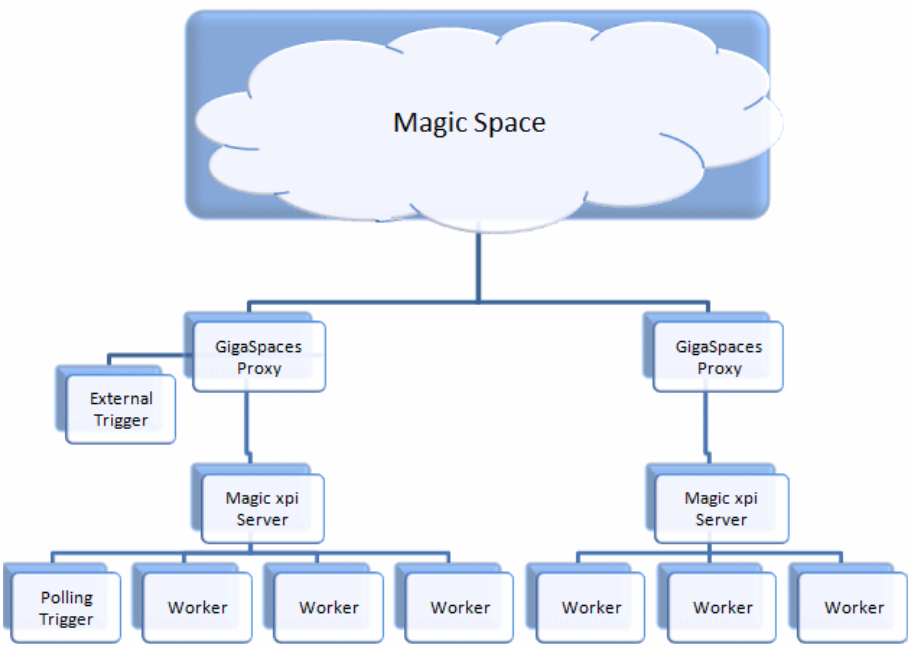
4. PU が START_REQUESTED のステータスを持つサーバエンティティを見つけたら、start.xml ファイルに基づいてサーバエンティティで定義されたホスト名または IP アドレスに従って、Grid Service Agent (GSA)をスキャンします。PU は、このサーバエンティティで定義されたすべてのパラメータ/起動情報を GSA に渡します。GSA が使用できない場合、PU は、それが利用可能かどうかを確認するために後の段階でチェックを続けます。
5. サーバエンティティで指定されたホストマシン上で実行されている GSA は、PU から渡されたパラメータに従って、Magic xpi サーバ(エンジンとも呼ばれる MgxpiServer.exe プロセス)を起動します。
    - プロジェクトが実行されていない場合、開始する最初の MgxpiServer.exe プロセスは、プロジェクトエンティティを Magic Space にロードしてプロジェクトを作成します。
    - プロジェクトが既に実行されている場合、ロードされた MgxpiServer.exe プロセスは既存のプロジェクトに参加し、独自のワーカーとトリガを作業フォースに追加します。
6. Magic xpi プロジェクトに渡されるパラメータの 1 つは、Space で作成されたサーバエンティティの ID です。Magic xpi サーバが起動すると、Magic Space に接続し、ID で識別されるサーバエンティティを検索し、ステータスを RUNNING に更新します。 プロジェクトの実行順序の詳細については、ここをクリックしてください。
7. サーバがリクエストを処理できるようになりました。

以下の方法でプロジェクトを停止します。:

-  リンクをクリックします。Stop リンクはプロジェクトのビルド時にプロジェクト フォルダに作成されます。
- モニタあるいはデバッガの Stop オプションをクリックします。

Magic xpi プロジェクト起動シーケンス

次の図は、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シーケンスを示しています。:



Magic xpi サーバ(エンジン)は、クライアントアプリケーションを Magic Space に接続する GigaSpaces プロキシ(ミドルウェアレイヤ)を通じて Magic Space と通信します。Magic Space は、複数のプロセスと複数のマシンを 1 つのユニットとして実装することができます。GigaSpaces プロキシは、サーバと Space パーティション間の正しい接続を担います。

1. プロジェクトが実行されているときは、ワーカー、ポーリングトリガー、外部トリガーの 3 つの要素が同時に動作します。 各 Magic xpi サーバは、さまざまなタスクに対して 1 つ以上のワーカー/トリガーを実行できます。:
    a. 非同期トリガーともよばれる(Directory Scanner のような)ポーリング トリガーはレスポンスを待たずにフローを起動する必要があるかどうかを確認し、外部システム(email アカウントなど)を常にチェックする Magic xpi サーバ スレッドです。フローを呼び出す必要があるときは、READY_FOR_USE というステータスのフローメッセージを Magic Space に書き込みます。
    b. 実行可能な各ワーカーはジョブが実行されるよう Magic Space をスキャンします。つまり、ステータスが READY_FOR_USE のメッセージをスキャンします。作業者がメッセージを見つけたら、ステータスを IN_PROCESS に変更し、

メッセージペイロードをフローに渡して、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)。

注: 2 つのワーカーが同じメッセージを実行するのを防ぐため、IN_PROCESS へのステータスの変更はトランザクション内で実行され、1 つのワーカーだけが行うことができます。

c. HTTP リクエスタや Web サーバなどの**外部**トリガーは外部のアプリケーションです。 トリガーは要求を受け取ると、Magic Space にフローを呼び出すメッセージを書き込みます。同期外部トリガーは、レスポンスメッセージも待機します。

2. メインフローが完了すると、ワーカーは Magic Space 内のリクエストメッセージのステータスを DONE に更新し、追加されたメッセージを自由にスキャンします。
3. メッセージが同期トリガー(HTTP トリガーなど)からのものである場合、フローが終了すると、ワーカーは Magic Space にレスポンスメッセージを書き込み、トリガーはクライアントに送信されます。
4. 全てのパラレルおよびスタンドアロン分岐は、Magic Space へのフローによって書き込まれた別個のメッセージとしても扱われます。 この新しいスレッド(パラレルまたはスタンドアロン分岐)は、プロジェクトのワーカーのいずれか、つまり別のマシン上で実行されているワーカーであっても処理できます。

Magic xpi プロジェクト停止シーケンス

適切なタイムアウト値を設定することで、プロジェクトを正常にシャットダウンすることができます。 シャットダウンコマンドが送信され、タイムアウトに達するまで、プロジェクトは新しい要求を処理しません(すべてのトリガーが停止します)。 しかし、プロジェクトは既に受け付けているリクエストは処理を継続します。

バッチから複数のプロジェクトを開始する

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

"D:\Magic xpi\Magic xpi 4 GS 11_12\MgxpiCmdl.bat" start-servers -startup-config-file "D:\Magic xpi\Magic xpi 4 GS 11_12\projects\Project1\start.xml" -Space-name "MAGIC_SPACE" -group "Magicxpi-4.0.0_AVIW-7-LP" -locators ""

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別設定の **start.xml** ファイルをロードできます。

GigaSpaces で使用されるポートの制御

LookupLocators 設定

discovery port(検出ポート)が GigaSpaces のデフォルトと異なる値に設定されている場合は、**LookupLocators** の値を変更して、LUS のホスト名または IP アドレスに加えて検出ポートの詳細を設定する必要があります。

Firewall 設定

最も一般的なシナリオは、全ての GigaSpaces エンティティがファイアウォールの背後あり、Web リクエスタまたは Web サービスリクエスタのみが DMZ 内にある構成です。ファイアウォールの設定方法の詳細については、Magic xpi ヘルプの**ファイアウォールの設定**を参照してください。

リカバリー動作

Magic xpi 4.9 は多様な障害シナリオから自動的にリカバリーすることができます。リカバリー・メカニズムは大別して 2 つのカテゴリに分類されます。:

物理的/グリッドの障害

物理的またはグリッドコンポーネントに障害が発生した場合、グリッドは完全に全自動でサービスをリカバリーするメカニズムを備えています。全てのグリッドサービスは中断なく機能し続け、少なくとも 1 台のマシンが稼動している限り、データは失われません。

Magic xpi ワーカーを実行しているマシンに障害が発生すると、エンジン/ワーカ障害のリカバリー メカニズムが起動されます。

エンジン/ワーカーの障害

リカバリー メカニズムは 3 つの要素から構成されます。:

識別

各ワーカーとエンジンは、そのステータスを Space に報告します。Magic PU はそのレポートをモニタします。タイムアウトに達していないワーカー/エンジンが見つかると:

- フローワーカー(トリガまたはフロー)の場合、ワーカを実行しているエンジンにワーカステータスを問い合わせます。

- エンジンの場合、エンジンがまだ生きている場合このエンジンを起動した GSA に問い合わせます。

エンジンとワーカーのリカバリー

応答しない、または存在しないエンジンの場合、GSA は、既存のエンジンを停止し、新しいエンジンを開始しようとします。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下に新たなワーカーを開始します。

ワーク プロセスのリカバリー

ワーク プロセスは、ルートフロー全体(その子フローを含む)の実行ツリーです。ワークプロセス全体は、1 つのビジネストランザクションとして扱われます。ルートフローとは、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、トリガー、スケジューラ、自動起動または Publish/Subscribe シナリオによって開始された可能性があります。

1. リカバリーは、個々のフローではなく、ワーク プロセスに対して定義されます。
2. Magic xpi は、ルートフローで定義されたリカバリーポリシーを使用し、子フローに対して定義されたリカバリーポリシーを無視します。

注: スタンドアロン分岐は、ワーク プロセスから切り離されているため、ワーク プロセスの一部とはみなされません。 したがって、独自の実行ツリーがあります。 スタンドアロン分岐がクラッシュした場合、リカバリー ポリシーはメインフローからではなく、独自のフロー プロパティから取得されます。

3. セーブ ポイントは最上位のフローのリニアの分岐にのみ保存されます。 子フローまたは子コンテキストで定義されたすべてのセーブポイントは無視されます。
4. ワーク プロセスは複数のワーカーで実行する事ができ(パラレル分岐の場合)、複数の別個の物理サーバ上で実行することができます。
5. ワーク プロセスの一部が実行されているワーカーがクラッシュした場合、全てのワーク プロセスはアボートし、最上位フローで定義したクリーンアップ フローが呼び出されます。
6. リカバリーが適用されると:
    a. アボート(Abort)の場合 – それ以上のアクションは不要です。
    b. 再起動 (Restart)の場合 – Magic xpi は起動されたペイロードで最上位のフローを再起動します。
    c. セーブポイント(Save Point)の場合 – Magic xpi は最終セーブポイントから起動されるか、セーブ ポイントが存在しない場合はフローが再起動されます。

トリガーのリカバリー

Magic xpi の各トリガーにはオプションのキープアライブ特性があります。この特性はトリガーが誤動作していると見做されるまでの待機時間と再起動を制御します。例外は外部トリガーである Web Services トリガーと HTTP トリガーです。これらのトリガーは外部プロセス(IIS/Apache/Systinet サーバ)上で動作し、個別に監視/管理を行うことができます。