

IBM WebSphere ESB ソフトウェアによる IT 柔軟性の向上

Beth Hutchison, Marc-Thomas Schmidt, Chris Vavra 執筆
日本アイ・ビー・エム ソフトウェア事業 翻訳

はじめに

現在、コストを抑えながらビジネス成長を発展させるために、多くの企業が既存IT資産の柔軟性の向上と再利用の方法を求めています。Service Oriented Architecture (SOA) は、明確な業界標準のインターフェースを使用して、例えば「顧客預金口座確認」といった何度も実行されるような処理をサービスとして定義する方法を提供します。エンタープライズ・サービス・バス (ESB) は、SOAにおいてサービスを統合するために使用される接続基盤を提供します。SOAとESBにより、インターフェースの数や複雑さを軽減することができ、企業はIT基盤の整備のためではなく中核となるビジネスの課題に焦点を置くことが可能となります。

このホワイト・ペーパーでは、IBM WebSphere Enterprise Service Bus (WebSphere ESB)ソフトウェアをご紹介します。ESBの活用によってどのような利点をもたらされるのかを説明します。この製品により、既存のアセットを再利用しながら新規のサービスを迅速に追加しサービスの変更を簡単に実施することができるESBの開発および実装が可能となります。

SOA と ESB

SOAは、新規のアプリケーションやサービスの構築のみならず、既存のアプリケーションやサービスの再利用または拡張において、柔軟性、拡張性のある構築手法を提供します。サービスとしては、IBM CICS、IBM IMS、Java 2 Platform, Enterprise Edition (J2EE)、Enterprise Java Beans (EJB)、Javaクラス、IBM DB2 Query、Microsoft .NETなど幅広い技術を実装可能です。SOA連携パターンでは、サービスを提供するサービス・プロバイダーは、実装するインターフェースや接続にあたっての規定（ポリシー）を宣言することにより、そのサービスが提供する機能を公開します。サービスを要求するサービス・リクエスターもまた必要なインターフェースやサポートする連携手段を宣言します。宣言はWeb Services Description Language (WSDL) や他のWebサービスにより規定されている標準手段により行われます。サービス・リクエスターは、サービス・プロバイダーでどのような実装が行われているかを意識することなくリクエストします。SOAはすなわち、サービスの実装部分をサービス定義および使用方法から分離することによってビジネス機能を仮想化する能力を提供します。

ESBはSOAにおける柔軟性を最大限にするための基盤としての役割を持ちます。サービス連携を行うサービス同士は互いに直接接続するのではなく、それぞれがESBに接続されます。サービス・リクエスターがESBに接続され、ESBはサービス・リクエスターから送信されたリクエストを、適切な機能と品質を持つサービス・プロバイダーに伝達する責任を担います。ESBは、リクエスターとプロバイダー間のプロトコルの違い、連携パターンや

サービスの機能の違いを吸収する手助けも行います。また、ESBによりモニタリングや管理機能も拡張可能です。すなわちESBはSOAのコア機能を実装し拡張する仮想機能や管理機能を提供します。

ESBにより以下の項目を仮想化することができます：

- ・ ロケーションと実体: 連携に参加するサービスは他のサービスのロケーションもしくは実体を知る必要がありません。例えばサービス・リクエスターは、複数あるうちのどのサービス・プロバイダーがリクエストを受信可能なのかを知る必要がなく、そのためサービス・プロバイダーは、サービス・リクエスターへ影響を及ぼさずにサービスの追加や削除が可能です。
- ・ 接続プロトコル: 連携に参加するサービスは共通のプロトコルもしくは接続方法をもつ必要がありません。SOAP/HTTPプロトコルを持つサービス・リクエスターからSOAP/JMSのみを受け付けるサービス・プロバイダーへの接続も可能です。
- ・ インターフェース: リクエスターとプロバイダーは共通のインターフェースをもつ必要がありません。ESBにより、サービス同士の要求および応答メッセージを相手のサービスが期待する形式に変換することが可能です。
- ・ サービスの品質(QoS): 参加するサービスもしくはシステム管理者にて、自らのサービスの品質の要件を宣言します。リクエストの権限、メッセージ・コンテンツの暗号化および復号化、サービス接続およびどのようにリクエストがルーティングされるかの自動監査（例：スピードもしくはコストの最適化）などです。

参加するサービスとサービスの間にはESBを介在させる際、ESB上での処理はメディエーションと呼ばれる論理的なモジュールにより構成されます。メディエーションはリクエスターとプロバイダー間の通過地点に位置づけられ、メッセージを操作します。複雑な連携の場合には、複数のメディエーションを連続して並べる構成も可能です。

SOAプログラミング・モデルにおける ESB

IBMは、サービスを実装し、またサービスをソリューションとして組み立てるためのプログラミング・モデルを発表しました。Service Component Architecture (SCA) とService Data Objects (SDO)はSOAプログラミング・モデルの基盤を提供します。SCAはサービス・コンポーネントを表現するモデルを定義し、それらをソリューションへと組み立てる方法を提供します。SDOはコンポーネント間でやりとりされる情報のモデルを定義します。SCAとSDOはWebサービス・スタンダードであるWSDLやXMLスキーマ定義 (XSD)言語をベースにしており、またSOAのコンポーネント・モデルを定義するために、それらの相互運用性標準を増強します。このモデルはIBM developerWorksというWeb記事に掲載されており、以下より参照可能です。

“Introduction to the IBM SOA programming model”

<http://ibm.com/developerworks/webservices/library/ws-soa-progmodel/>

WebSphere ESBは、SCAにより表現されるエンドポイント間のメッセージのフローを管理し、各コンポーネントが要求する接続時の品質を管理します。ESB上のメディエーションはルーティングやロギングの機能、リクエスターとプロバイダー間での、プロトコル、連携スタイル、インターフェース、などの違いを吸収します。SCAをベースとするソリューションの中で、メディエーションは特別な役割を有するSCAコンポーネント・パターンを使用して実装されます。したがって、ビジネスに関連する処理を扱う他のコンポーネントとは若干異なる性質を持ちます。メディエーション・コンポーネントはサービス・エンドポイント間のメッセージ交換を扱います。通常のビジネス・アプリケーション・コンポーネントとは対照的に、メディエーション・コンポーネントは、インフラストラクチャー上のメッセージのフローに関係しており、メッセージのビジネス・コンテンツのみ扱うわけではありません。ESBは、いわゆるビジネス機能ではなく、メッセージのルーティング、変換、ロギングなどを扱います。メッセージの挙動を制御する情報は、しばしば、ビジネス・メッセージとともに流れるヘッダーに含まれます。IBM SOAプログラミング・モデルは、そのパターンをサポートするために、SDOにService Message Object (SMO)パターンを提供しています。SCAに関する詳細情報は下記より参照可能です。

www.ibm.com/developerworks/library/specification/ws-sca/

WebSphere ESB と WebSphere Integration Developer

WebSphere ESB は、IBM SOA プログラミング・モデル環境において ESB 機能を提供します。WebSphere ESB は異なるプロトコルと通信 (SOAP/HTTP, SOAP/JMS, JMS, J2EE Connector Architecture (JCA) Adapters など) を利用してサービス・エンドポイント間の通信を実現します。また、WebSphere ESB のメディエーション機能としてこれらのエンドポイント間でのメッセージ変換、ロギング、ルーティングを行います。

WebSphere ESBのメディエーションの作成は、IBM WebSphere Integration Developer という開発ツールを使用して、様々なビジネス・コンポーネントとの組み合わせによるSCAソリューションを構築することで実現します。WebSphere Integration DeveloperによりWebSphere ESBに事前定義されたメディエーション・コンポーネントを使用することができ、これにより開発時間の削減が実現可能となります。また、アプリケーションに特化したアダプター機能も提供されています。WebSphere ESBとWebSphere Integration Developerを活用することにより以下の4つのメリットが提供可能であり、このホワイト・ペーパーではこれよりこれらについて言及していきます。

- Web サービス接続、メッセージングおよびサービス指向連携
- ESB ベースのメディエーションのアセンブリーから、テスト、デプロイ、管理まで

- のライフサイクル管理による使い易さの向上
- ・ 市場投入への時間短縮
- ・ WebSphere ソフトウェア製品群とのシームレスな連携

Web サービス接続、メッセージングおよびサービス指向連携

WebSphere ESBはサービス・エンドポイント同士の連携を 3 つのレベルでサポートします。多種多様な接続性、連携モデルと品質の多様性、そしてメディエーション機能です。WebSphere ESBは、様々なプロトコルやアプリケーション・プログラミング・インターフェース (API) をサポートします。たとえば、WebSphereプラットフォーム・メッセージングとして実装されている JMS V1.1 や IBM WebSphere MQ、また SOAP/HTTP や SOAP/HTTPS、SOAP/JMS などです。WebSphere ESB V6.0.2 ではパフォーマンスや接続性が大幅に改善されています。

WebSphere ESB は、WebSphere Application Server 上にビルドされていることから、WebSphere ポートフォリオである WebSphere MQ や WebSphere Message Broker との親和性もあります。最新のリリースでは、ネイティブ WebSphere MQ バインディングもサポートされるようになり、WebSphere MQ や WebSphere Message Broker との連携がこれまで以上に容易にまた早急に実現できるようになりました。WebSphere ESB はまた、WebSphere Adapter ソリューションを利用した既存アプリケーション資産との連携により、幅広い範囲でビジネス・イベントを扱うことも可能となっています。WebSphere Integration Developer により、4 つのテクノロジー・アダプター (Email、FlatFile、File Transfer Protocol (FTP)、Java Database Connectivity (JDBC))を開発および本番局面で使用可能に、また 4 つの JCA enterprise resource planning (ERP) アダプター (SAP、PeopleSoft、Oracle E-Business and JD Edwards) を開発局面で使用可能です。

WebSphere ESBが持つクライアント・インターフェースにより接続性を更に拡張することが可能です。Message Service Clients for C/C++ and .NETにより非Java実装のアプリケーションがJMS APIに似たAPIを使用してWebSphere ESBへ接続可能となります。

Web Services Client for C++は Java API for XML-based Remote Procedure Call (JAX-RPC)に類似しており、ユーザーに対し WebSphere Application Server 上の Web サービスに C++環境から接続が可能となります。接続性のその他の機能として、リクエスターがリクエストを投げる際のエンドポイントのプロトコル(たとえば、SOAP/HTTP)が、実際にこれらのリクエストを処理するサービス・プロバイダーのプロトコル(たとえば、SOAP/JMS)と異なる場合の基本的なプロトコル変換を実施します。

WebSphere ESBはさまざまな連携モデルをサポートします。リクエスト/リプライ、Point

to Point、パブリッシュ/サブスクライブなどです。またWebSphere ESBはWebサービス標準であるWS-Security、WS-Atomic Transactionもサポートします。また、Universal Description, Discovery and Integration (UDDI) V3.0 というサービス・エンドポイント・メタデータを公開し管理する機能によりクライアント・アプリケーションにサービス定義情報を提供します。統合開発者は、メディエーション・モジュールを開発する際にUDDIを参照してインターフェースのありかを検索、提供します。WebSphere ESBはまたIBM WebSphere Service Registry and Repositoryもサポートします。WebSphere Service Registry and Repositoryと連携することで、WebSphere ESBのダイナミックエンドポイント・ルックアップ・プリミティブによりサービス・エンドポイントの情報を問い合わせる新規のサービス・プロバイダーへの接続することを、再開発や再デプロイを行う必要なく実施することが可能です。WebSphere Service Registry and Repositoryとの連携により、ポリシーによる統治で最適化されたランタイムアクセスを伴うサービス利用の柔軟性をシステムに与えます。

最後に、WebSphere ESB は、接続機能でサポートされるプロトコルの変換より上位の処理として、サービス・エンドポイント間でのメディエーション機能をサポートします。この機能により連携ロジックはサービス・エンドポイントではなく、ESB において実現されます。WebSphere ESB メディエーション機能では、コンテキスト・ベース、また他の形式でのメッセージング・ルーティング、ロギング、メッセージ変換などが提供されています。

使いやすさの向上

WebSphere ESBとWebSphere Integration Developerは統合的なドキュメンテーション、わかりやすいサンプルにより、ユーザーが即座に使用できるようにデザインされています。WebSphere Integration Developerは、これまで統合開発者がESBベースのソリューションを設計、テスト、構成、デプロイする際に必要であったプログラミング・スキルが最低限で済むようデザインされた使いやすいツールとなっています。統合管理者は、グラフィカルなツールによりサービス・エンドポイントを定義し接続できるようになっており、これらを任意にメディエーション・フローに接続可能なようにビルドできるようになっています。メディエーション・フローではパレットからメディエーション・プリミティブを選択して、構成し、互いを接続するといった処理をグラフィカルに行えます。これらのプリミティブにはメッセージのルーティング、コンテンツの品質向上、ロギング、変換などがあります。統合開発者はWebSphere ESBランタイム環境にモジュールをデプロイする前にWebSphere Integration Developerの環境にてローカルのテストを実施しデバッグを行います。ランタイムでは、提供されているWebSphere ESB管理コンソールにより、ソリューション管理者が容易にWebSphere ESBを管理することができるようになります。

市場投入への時間短縮

WebSphere ESB は、既存の IT 資産を有効利用しながら柔軟な連携インフラを構築することによってサービス連携を実現するために、容易で費用対効果の大きいソリューションを提供します。WebSphere ESB がサポートする様々なビジネスおよび IT 標準により更なる接続性や移植性が促進され、JCA ベースのアダプターを含む WebSphere アダプターによって、多数の独立系ソフトウェア・ベンダー(Independent Software vendor: ISV)ソリューションのサポートを増やします。

WebSphere ESB では、構築されたビジネス・プロセスを効果的に再構築し管理することも可能です。統合開発者もしくはシステム管理者において、周囲の ESB ベースのソリューションに影響を与えずに動的なエンドポイント変更や追加を実現可能です。また、Common-event infrastructure (CEI)サポートにより、モニタリングを行い特定のイベント発行を行って WebSphere Business Monitor と連携することが可能です。WebSphere ESB 管理コンソールでは基盤となる WebSphere Application Server の管理機能がすべて提供されるため、WebSphere Application Server の機能もすべて活用可能です。

WebSphere ソフトウェア製品群とのシームレスな連携

WebSphere ESB はベースとなる IBM WebSphere Application Server Network Deploymentプラットフォームの機能をすべて使用可能なため、Quality of Service (QoS)、ワークロード・バランシング、クラスタリング、高可用性、スケーラビリティなどWebSphere Application Serverの持つ機能を継承します。WebSphere Application Serverとの深いつながりにより、WebSphere ESBは、IBM Tivoliセキュリティ、ディレクトリおよびシステム管理機能、つまり、IBM Tivoli Access Manager (高度なセキュリティや統合されたパーソナライズド・エクスペリエンス)や、IBM Tivoli Directory (Lightweight Directory Access Protocol [LDAP] サーバーとして使用可能)などの連携も可能です。WebSphere ESBはまた IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA)との連携により WebサービスのメッセージのモニタリングやWebサービスのエンドポイントの管理が可能となります。Tivoli Composite Application Manager for SOAはサービスのパフォーマンス情報をWebSphere Service Registry and Repositoryへ渡し、WebSphere ESBの動的エンドポイント選択の制御を提供します。

WebSphere ESB は WebSphere Application Server と WebSphere Process Server と共通の管理コンソールを持つことで、WebSphere 製品ファミリーをまたがった共通オペレーション制御のインターフェースとなっているため、これらすべての管理を一元的に行うことが可能となります。また、WebSphere ESB は既存の WebSphere MQ メッセージング基

盤と一体化させることで、既存のメッセージング・バックボーンを、オープン・スタンダードを使用した環境として実装することが可能です。また、WebSphere ESB は WebSphere Message Broker と連携が可能なので、WebSphere ESB がスタンダード・ベースの Web サービス相互接続を行い、WebSphere Message Broker にて多種多様なメッセージ・フォーマットを扱うことによって、複雑な ESB トポロジーを実装することもできます。

ESB要件が増え、ビジネス・プロセスやステート・マシンによるエンドポイントのオーケストレーションが必要になったり、ビジネス・ルールによる動的な意思決定など、より高度な連携機能が必要となった際には、WebSphere ESBをWebSphere Process Serverへアップグレードすることが可能です。統合開発者はWebSphere Integration Developerでどちらのランタイムも使用可能ですので、必要に応じて開発環境を拡張できます。同様に、管理コンソールもWebSphere Application Serverから、WebSphere ESB、WebSphere Process Serverへと拡大することも可能です。

WebSphere Integration Developer による ESB ソリューションの構築および管理

ESB のユーザーの役割と作業

IBM では、ESB ベースのソリューションの構築面と管理面における 2 つのユーザー役割を紹介しています：

- ・ 統合開発者(Integration Developer) は、ESB 関連ツールとテクノロジーを使い、エンドポイントの定義を行い、サービス間でリクエストがどのようにルーティングされるかを制御するロジックを構築し接続します。この役割を担う担当者は連携されるビジネス・サービスの動作を理解しており、これらの連携を実現するためのメディエーション・モジュールを作成することに焦点を当てます。統合開発者は、開発ツールとして WebSphere Integration Developer を使用します。
- ・ ソリューション管理者は必要な新規サービスをデプロイしたり、既存のサービスと新規サービスを連携するメディエーションをデプロイすることで新たな SOA ソリューションを作成します。この役割の担当者は組織のビジネス・プロセス同士の基本的な接続パターンや全体的なソリューションを理解している必要があります。ソリューション管理者はデプロイされたソリューションの構成を変更することができ、IT システムのオペレーターによりモニタリングされた結果に応じて変更を行います。ソリューション管理者は、WebSphere Application Server および WebSphere ESB の管理コンソールで提供されている機能を使用します。

メディエーション機能の作成はエンドポイントを効果的に作用させる必要がある

統合開発者はWebSphere Integration Developerを使用して、連携されるエンドポイント、および接続に使用されるバインディングや接続プロトコルを明記するためにメディエーション・モジュールを作成します。メディエーション・モジュールを通過するメッセージに対して行う必要のある処理はメディエーション・プリミティブを選択し組み立てることにより定義します。WebSphere Integration Developerは、下記の事前定義されたメディエーション・プリミティブをパレット上に提供します：

- ・ フェイル：例外を発生してメディエーション・フローの処理を停止します
- ・ ストップ：暗黙的にメディエーション・フローの処理を停止します
- ・ メッセージ・エレメント・セッター：カスタム・コーディングを行わずに、単純なデータ変換を実施します
- ・ メッセージ・フィルター：XPath を使用してメッセージのコンテンツを比較し、結果に応じて転送する後続メディエーション・プリミティブにメッセージをルーティングします
- ・ サービス・レジストリー・ルックアップ：WebSphere Service Registry and Repository よりエンドポイントを選択します
- ・ Extensible Stylesheet Language Transformations (XSLT)：XSL スタイルシートの定義に基づきメッセージを変換します
- ・ データベース・ルックアップ：特定の値をデータベースから取得し、XPath式を使用して、メッセージの1エレメントとして格納します
- ・ メッセージ・ロガー：監査証跡などの目的のためにメッセージのコンテンツを XML としてデータベースにロギングします
- ・ イベント・エミッター：メディエーションから Common Base Event を生成します (WebSphere Business Monitor のモニタリング対象として使用します)

統合開発者はこれらメッセージ・プリミティブをカスタマイズします。たとえば、検索すべきデータベースを指定したり、XSL スタイルシートを提供するといった使い方をします。

これらのメディエーションを構成する際に、プログラミングのスキルは必要ありません。それは、WebSphere Integration DeveloperがWSDL、XMLスキーマ、XPath、XSLTなどの複雑さを隠蔽しているためで、これにより統合開発者はこれらのESBのコア技術スキルがなくとも、十分にSOAソリューションを構築することができます。提供されているプリミティブで要件を満たすことができない場合は、SCA Javaコンポーネントを使用してカスタム・メディエーション・プリミティブを使い、Javaコードを直接もしくはビジュアルに記述することも可能です。また、メディエーションを構築するために使用することができる全てのSCAやSDOプログラミングAPIやSystem Programming Interface (SPI)機能も提

供されています。

WebSphere Integration Developer を使用することにより、同期、非同期共にリクエスト/レスポンス形式のメッセージ・フローの記述も可能です。レスポンス・フローはリクエスト・フローの作成方法と同様の手順で作成します。一連のフローが完成したら、ビジュアル・デバッガー機能を使用してメディエーションのリクエスト・フロー、レスポンス・フローにブレイクポイントを設置しステップごとにフローを実行したり、メッセージのエレメントの内容を確認することができます。

メディエーション・モジュールのデプロイおよび管理

メディエーション・モジュールのデプロイは **WebSphere ESB** のデプロイツールであるサービス・デプロイ機能により行います。**WebSphere ESB** の管理は **WebSphere Application Server** の管理コンソールがベースとなっているため、基盤のアプリケーション・サーバーと統合された共通のインターフェースにより **ESB** を管理するためのすべての機能を持っています。なお、ソリューション管理者は、「アプリケーションの統合」タスク・フィルターを使用して、役割に関連した情報だけを管理コンソール上に表示させるよう選択することも可能です。また、ソリューション管理者はいつでもすべての機能を表示する画面に戻ることもできます。

メディエーション・モジュールは **WebSphere Application Server** 上の他の成果物と共存が可能で、個々のパフォーマンスを **WebSphere ESB** の一部である **IBM Tivoli Performance Monitor** からモニタリングすることが可能です。また、**Web** サービス・バインディングにより接続されたサービスやメディエーションについては **IBM Tivoli Composite Application Manager for SOA** を使用して **Web** サービス間を **ESB** を通じて流れるメッセージをトラッキングすることや、メッセージのスループット、応答タイムをモニタリングすることが可能で、必要に応じてアラートの発行なども行えます。

結論

SOA は **IT** アーキテクチャーの次なる進化へのステップとして企業が複雑なチャレンジに直面したときに既存の開発者、ソフトウェア言語、ハードウェア・プラットフォーム、データベース、アプリケーションなどを活用しながら費用とリスクを最小限に抑えつつ生産性を上げることを実現します。この順応性のある、柔軟なアーキテクチャーにより、市場へのより短時間での対応や開発とメンテナンスにおけるコストやリスクの軽減を実現します。

これまでの歴史を見ても、アプリケーションは古くなったからといってすぐには消滅す

るわけではありません。アプリケーションは古くなっても、システムとして稼動している限り生き残ります。結果として既存のアプリケーションと新しいアプリケーションを連携する際に、データの変換、ルーティングなどを柔軟に実施する ESB の役割はきわめて重要です。WebSphere ESB は SOA プログラミング・モデルの環境において ESB の役割を実現する製品です。WebSphere ESB により、幅広いプロトコル変換、エンドポイント間のメッセージ変換、ロギング、ルーティングを実施してサービス・エンドポイント間の連携を促進します。

WebSphere ESB の開発ツールである WebSphere Integration Developer により、プログラミング・スキルが最小限で済む統合的なビジュアルな開発が提供されます。統合開発者は理解しやすいドキュメンテーションと用意されたサンプルを使用して開発スキルを習得できます。開発はシンプルに実施できる構成となっており、サービスを宣言してサービスの接続方法を定義し、ツールに提供された機能により、メッセージ・ルーティングや変換を行うメディエーションをビジュアルに定義できます。また、役割ベースの管理により、ソリューション管理者に分かりやすい形で WebSphere ESB を管理できるようになっています。

WebSphere ESB により、時間短縮も実現できます。サービス連携のための投資対効果のあるソリューションとして、他ベンダーも含む既存の資産価値を拡張させる可能性のある柔軟な連携インフラストラクチャーを構築することで WebSphere ESB は SOA IT 資産を有効活用させます。多数の ISV ソリューションや WebSphere アダプターをサポートすることで、既存資産との連携も容易に実現できます。事前定義されたメディエーション機能により、開発に費やす時間やコストを軽減させます。また、WebSphere ESB が WebSphere Application Server をベースとしていることから、サービス品質であるクラスタリング、フェイルオーバー、システム管理やセキュリティを活用することもできます。共通のツールと管理ということで、WebSphere ESB から WebSphere Process Server へのシームレスな変更も実現可能です。また Tivoli ソフトウェアとの連携により、ワールドワイドで認められたセキュリティとシステム管理機能も提供できます。これらの 4 つのキー・バリューにより、WebSphere ESB ソフトウェアはビジネスを SOA 化させる役割を担います。

詳細な情報

SOA プログラミング・モデルについて更に学びたい場合は、以下 URL をご参照ください。
ibm.com/developerworks/library/ws-soa-progmodel4/

IBM WebSphere ESB について更に学びたい場合は、IBM 担当者、IBM ビジネス・パートナーまでご連絡いただくか、以下 URL をご参照ください。

ibm.com/software/integration/wsesb/

IBM WebSphere Integration Developer について更に学びたい場合は、IBM 担当者、IBM ビジネス・パートナーまでご連絡いただくか、以下 URL をご参照ください。

ibm.com/software/integration/wid/

グローバル WebSphere コミュニティに参加される場合は、以下 URL をご参照ください。

www.websphere.org