IBM® Information Management Software

Information Server デプロイメント・アーキテクチャー

- 小規模から始めて大規模に拡張する道のり
- メタデータに関する考慮事項について
- グリッドおよび SMP 環境でのデプロイメント

Chuck Ballard
Tuvia Alon
Naveen Dronavalli
Stephen Jennings
Mark Lee
Sachiko Toratani

ibm.com/redbooks

Redbooks

International Technical Support Organization

Information Server
デプロイメント・アーキテクチャー

2012 年 12 月

SG88-4072-00
（英文原典 :SG24-8028-00）

お願い : 本書および本書で紹介する製品をご使用になる前に、vii ページの『特記事項』の情報をお読みください。

第 1 版 (2012 年 12 月)

本書は、IBM InfoSphere Information Server のバージョン 8.7 に適用されます。

目次

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾 :

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。これらおよび他の IBM 商標に、この情報の最初に現れる個所で商標表示 (® または ™) が付されている場合、これらの表示は、この情報が公開された時点で、米国において、IBM が所有する登録商標またはコモン・ロー上の商標であることを示しています。このような商標は、その他の国においても登録商標またはコモン・ロー上の商標である可能性があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、International Business Machines Corporation の米国およびその他の国における商標です。

AIX®
ClearCase®
Cognos®
DataStage®
DB2 Universal Database™
DB2®
GPFS™
Guardium®
HACMP™
IA®
IBM®
IMS™
Information Agenda®
Informix®
InfoSphere®
Optim™
PowerHA®
QualityStage®
Rational Team Concert™
Rational®
Redbooks®
Redbooks (ロゴ) ®
Tivoli®
WebSphere®
z/OS®

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft、Windows および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

序文

IBM® InfoSphere® Information Server (Information Server) は、異機種混合システムで信頼できる情報を配信するための包括的なメタデータ主導型プラットフォームを提供します。Information Server は、再利用可能な共有コンポーネント (コネクター、データ品質、複雑な変換など)、および共通のメタデータ・プラットフォームを使用するルールを通じてデータのプロファイル作成、データ統合、およびデータ配信を容易に行えるようにするスケーラブルなパラレル・エンジンを備えています。

Information Server は、ビジネス目標と IT アクティビティーを連携させる上で役立つ独自のメタデータ主導型の設計となっています。一貫性のある用語とルールを提供して、ビジネス仕様を収集します。これらの仕様を使用して開発作業を自動化し、データの来歴を追跡することによってデータからさらに大きな洞察を得られるようにします。

典型的なデプロイメント・アーキテクチャーでは、複数の製品、環境、およびサーバーにまたがる共有メタデータ・プラットフォームをフルに活用する上で課題が生じます。また、データ・プライバシーと機密保護の要件により、さらに複雑さの度合いが高くなります。

この IBM Redbook 資料では、このプラットフォームの共有メタデータ機能を使用する典型的な論理インフラストラクチャー・トポロジーで Information Server コンポーネントの導入を成功させるためのガイドラインと基準について説明します。これらのガイドラインは、開発ライフサイクル、データ・プライバシー、機密保護、高可用性、およびパフォーマンスに対する要件をサポートします。本書の目的は、お客様が、これらのガイドラインに基づいて独自の情報要件を評価し、適切な導入アーキテクチャーを判断できるように支援することです。具体的なユース・ケースを実行したり、ビジネス目標達成を成功させるために Information Server の製品モジュールとコンポーネントの機能を効果的に使用したりする上で役立ちます。

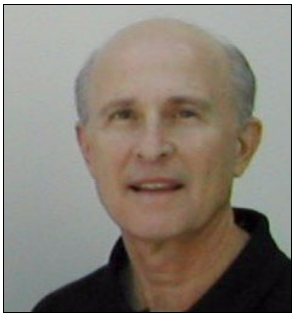
本書は、IBM InfoSphere Information Server の情報統合機能の完全スイートを提供する責任がある IT アーキテクト、情報管理と情報統合のスペシャリスト、およびシステム管理者を対象としています。

本書は以下の各章で構成されています。

- 第 1 章 : Information Server の概要。この章では、IBM InfoSphere Information Server のアーキテクチャーとその機能および製品コンポーネントの概要を説明しています。
- 第 2 章 : Information Server のインストール・トポロジー。この章では、基本的な 1 台のコンピューターから可用性の高いクラスターおよびグリッド構成に及ぶ Information Server のインストール・トポロジーについて説明しています。
- 第 3 章 : メタデータ管理。メタデータ管理とは、組織が「どのようにしたら、データに関する自分たちの理解内容を把握できるか」という質問に答えられるように支援するツール、プロセス、および環境を指します。この章は、メタデータを定義して、メタデータのタイプ、メタデータが重要である理由、および Information Server メタデータを表示できる場所を理解する上で役立ちます。
- 第 4 章 : Information Server 資産のライフサイクル管理。この章では、IBM InfoSphere Information Server Manager および外部ソース・コード管理システムを使用して IBM InfoSphere DataStage® および IBM QualityStage® アプリケーションのソース・コード管理管理を含むライフサイクル管理を実装する方法について説明します。また、この章では、IBM InfoSphere Information Analyzer、IBM InfoSphere Business Glossary、IBM InfoSphere Metadata Workbench、および関連するメタデータ・デプロイメント・アーキテクチャーのライフサイクル管理についても説明しています。
- 第 5 章 : メタデータのデプロイメント・アーキテクチャー。Information Server 製品モジュールを効果的に使用してビジネス目標を達成するには、Information Server のデプロイメント・アーキテクチャーを選択する際に多様なメトリックを考慮する必要があります。この章では、SIMPLE、UNIFIED、REPORTING、および GOVERNANCE のデプロイメント・アーキテクチャーを紹介しています。
- 第 6 章 : デプロイメントのユース・ケース。この章には、Information Server デプロイメント・アーキテクチャーをテストおよび検証するための多数のユース・ケースが記載されています。

本書の執筆チーム

本書は、世界の各地で International Technical Support Organization (カリフォルニア州サンノゼ) の業務に従事している専門家のチームが執筆したものです。



Chuck Ballard は、International Technical Support Organization (カリフォルニア州サンノゼ) のプロジェクト・マネージャーです。35 年を超える経験を積んでおり、製品エンジニアリング、販売、マーケティング、技術支援、および管理の分野における職務に従事してきました。データベース、データ管理、データウェアハウジング、ビジネス・インテリジェンス、およびプロセス・リエンジニアリングの分野を専門としています。これらのテーマについて幅広く執筆しており、講義を行い、世界中の会議やセミナーに参加しています。Chuck は、Purdue University の生産工学の学士号と修士号の両方を取得しています。

Tuvia Alon は、Information Management の IBM Software Group Lab Services Center of Excellence のシニア・コンサルタント兼ソリューション・アーキテクトであり、EMEA およびその他の世界中の地域にサービスを提供し、IBM Israel Software Lab に勤務しています。Tuvia は、2006 年、メタデータおよびセマンティックのテクノロジーを専門とする Unicorn の買収の際に IBM に入社しました。Tuvia は、エンタープライズ・ソフトウェアの開発、実装、およびサポートに 20 年以上携わってきました。メタデータ・テクノロジーの専門家であり、エンタープライズ・データ・ガバナンス・ソリューションの実装におけるリーダーとして認められています。Tuvia は、Center of Excellence のコンサルタントとして、データ統合およびデータ・ガバナンス戦略を実装する IBM のお客様の信頼できるアドバイザーとしての役割を果たしています。米国およびヨーロッパで、企業ガバナンス・ポリシーにおけるメタデータの使用に関する多数の会議のプレゼンターを務めています。最近、IBM Redbook「*Metadata Management with IBM InfoSphere Information Server*」(SG24-7939) を共同執筆しました。

Naveen Dronavalli は、IBM Information Management Group の Virtual Services チームのテクニカル・ソリューション・アーキテクトです。8 年間以上にわたり情報技術の分野に携わってきた経験があり、データウェアハウジングおよびデータ統合を専門としています。Naveen は、IBM Information Server Suite、データベース、およびオペレーティング・システムなどのツールに熟達しています。ETL およびデータ統合プロセスのアーキテクチャー設計の技術リーダーを務めたことがあり、高可用性を利用する Information Server on Grid のインフラストラクチャー・ソリューション・アーキテクチャーを提供しています。また、お客様向けのガイダンス、教育研修、およびトレーニングも行っています。Naveen は、Kuvempu University (インド) でエレクトロニクスおよび通信エンジニアリングの学士号、Illinois Institute of Technology で電気工学専攻の修士号を取得しています。

Stephen Jennings は、Information Management の IBM クライアント・テクニカル・スペシャリストであり、以前は IBM テクニカル・ソリューション・アーキテクトを務めていました。Information Server を使用するデータ統合およびデータ・ガバナンスのプロジェクトでお客様を 10 年以上支援してきた . 経験があります。Steve は、University of Louisville を卒業し、米国ケンタッキー州ルイビルに住んでいます。

Mark Lee は、IBM Software Group 内のシニア・テクニカル・コンサルタントです。20 年以上にわたり、幅広い IBM のお客様と協業して、複雑なデータ統合および並列処理ソリューションの実装を支援してきました。Mark は、University of Manchester のコンピューター・サイエンスの優等学位と修士号を取得しています。英国サリー州に住んでいます。

Sachiko Toratani は、IBM Japan の IBM InfoSphere 製品の IT スペシャリストです。10 年以上にわたり、政府関連システムの DBMS およびアプリケーション開発に携わってきた経験があります。専門分野は情報統合および DBMS であり、InfoSphere Information Server および DataStage に関する幅広いスキルを持っています。IBM の Database Administrator IBM DB2® Universal Databaseô for Linux, UNIX, and Windows の認定資格を保有しています。また、IBM Redbook「*IBM InfoSphere DataStage データ・フローとジョブ・デザイン*」(SG88-4045) および「*Deploying a Grid Solution with the IBM InfoSphere Information Server*」(SG24-7625) の制作にも貢献しました。

その他の貢献者

本書に対して文章、各分野の専門知識、およびご支援という形でご協力いただいた以下の方々に感謝いたします。

本プロジェクトに技術的専門家として携わってくださった以下の方々に感謝いたします。

- Paul Christensen: IBM Information Agenda® Architect、IBM Software Group、Information Management (米国ペンシルベニア州ウェスト・チェスター)
- Robert Johnston: Information Agenda Architect、IBM Software Group、Information Management (米国オハイオ州クリーブランド)

文章を提供してくださった以下の方々に感謝いたします。

- Julius Lerm: Advanced Consulting Engineer and Technical Solution Architect、IBM Software Group、Information Management (米国イリノイ州シカゴ)

本プロジェクトに貢献してくださった以下の方々に感謝いたします。

- Tony Curcio: InfoSphere Product Management、Software Product Manager (米国ノースカロライナ州シャーロット)
- Marc Haber: Functional Architect、Metadata Tools、IBM Software Group、Information Management (イスラエル、ハイファ)
- Ernie Ostic: Client Technical Specialist、IBM Software Group、Worldwide Sales Enablement (米国ニュージャージー州ピスカタウェイ)

International Technical Support Organization の以下の方々に感謝いたします。

- Mary Comianos: 出版物管理
- Emma Jacobs: グラフィックス・サポート
- Ann Lund: 研修管理
- Diane Sherman: 編集者

皆様も出版物の執筆にご協力ください

お客様のスキルにスポットライトを当て、キャリアを発展させながら、出版物の執筆者になることができます。ITSO の研修プロジェクトに参加して、お客様の専門分野を取り上げた書籍を執筆しながら、最先端のテクノロジーを実地体験することができます。お客様のご協力は、製品採用率やお客様の満足度の向上につながります。同時に、技術者との連絡ネットワークや人脈を広げることもできます。研修プログラムは 2 週間から 6 週間にわたって実施され、直接的に参加することも、ご自宅からリモート研修生として参加することもできます。

研修プログラムの詳細情報、研修のインデックスの閲覧、オンラインでのお申し込みについては、以下の URL にアクセスしてください。

ibm.com/redbooks/residencies.html

ご意見をお寄せください

皆様からの貴重なご意見をお待ちしております。

弊社では、できるだけ有益な書籍を提供したいと考えております。本書またはその他の IBM Redbook 資料に関するご意見を以下のいずれかの方法でお寄せください。

- 次のサイトにあるオンラインの Redbook レビュー・フォームの「**Contact us**」を使用する。

  ibm.com/redbooks

- ご意見を E メールで次のアドレスまで送信する。

  redbooks@us.ibm.com

- ご意見を次の住所まで郵送する。

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

IBM Redbook とつながりましょう

- Facebook で検索してください。

  http://www.facebook.com/IBMRedbooks

- Twitter でフォローしてください。

  http://twitter.com/ibmredbooks

- LinkedIn で検索してください。

  http://www.linkedin.com/groups?home=&gid=2130806

- 毎週発行される IBM Redbook のニュースレターで最新の Redbook 資料、研修、およびワークショップをご検討ください。

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- RSS フィードで最新の Redbook 資料の情報を入手してください。

  http://www.redbooks.ibm.com/rss.html

1

Information Server の概要

IBM InfoSphere Information Server (Information Server) は、統一されたデータ統合プラットフォームを提供する製品ファミリーであり、お客様が、信頼性が高くコンテキストに合った情報を理解して、情報のクレンジング、変換、および配信を行い、重要なビジネス・イニシアチブで活用できるようにします。Information Server には、プロセス全体を通じたガバナンスの機構など、統合されたエンドツーエンドの情報統合ソリューションを提供する多様な製品モジュールおよびコンポーネントが組み込まれています。Information Server プラットフォームの主要エレメントは強力な共有メタデータ層であり、ビジネス領域と技術領域の両方に対する可視性、レポーティングの基盤、および複数の製品モジュールにわたる統合ポイントを提供します。

Information Server は、組織がこのプラットフォームの利点を最大限に活用してビジネス目標を達成できるようにする柔軟なアーキテクチャーを提供します。成功する製品モジュールのデプロイメント戦略を判別するには、複雑な情報統合環境における多くのパラメーターを考慮する必要があります。

本書の究極的な目標は、お客様が独自の情報要件を評価する上で支援することです。本書は、記載されているガイドラインに基づいて、具体的なユース・ケースを実現できる適切なデプロイメント・アーキテクチャーを決定すると同時に、Information Server 製品モジュールおよびコンポーネントの機能を効果的に使用してビジネス目標の達成を成功させる上で支援します。そのために、本書では、Information Server に組み込まれている開発製品モジュールの代表的なライフサイクル・プロセスを示して、さまざまな開発段階を通じて開発コードが管理およびプロモートされる方法を説明し、これらの環境をサポートする複数のアーキテクチャー・インフラストラクチャーを示しています。

ただし、コードは、情報統合プロジェクトにおいてビジネスにもたらされる最終成果物ではありません。最終成果物は、抽出・変換・ロード (ETL)、およびクレンジングと強化のプロセスから得られる情報です。これらのプロセスにより、その情報によって実現できる意思決定のスピード、正確性、および価値が改善されます。この情報は、データと、データを記述してデータに関連付けられるメタデータの両方で構成されます。

データ管理は、本書に記載されている参考資料を含めて、多くの資料で取り上げられている高度な研究分野です。ただし、本書では、メタデータがどのように Information Server 製品モジュールによって生成され、製品モジュール間の統合を可能にするのか、メタデータの管理方法が開発資産とどのように異なるのか、したがってメタデータの利点を最大限に活用するためにアーキテクチャーをどのように設計すべきかについても説明しています。

本書では、最初に Information Server の概要を紹介するために各コンポーネントについて説明します。本章では以下のトピックを扱います。

- Information Server の紹介と機能の概要
- Information Server プラットフォームの機能層のインフラストラクチャー
- Information Server 製品モジュールおよびコンポーネント

バージョン 8.7: 特に断りのない限り、大半の方法論、プロセス、およびアーキテクチャーの説明は、Information Server の別個のバージョンに適用可能です。ベースラインとして Information Server バージョン 8.7 が使用されています。

1.1 概要

IBM InfoSphere Information Server は、企業が信頼性が高くコンテキストに合った情報を理解して、情報のクレンジング、変換、および配信を行えるようにする、統一された単一のプラットフォームを提供します。このプラットフォームは、主に情報統合領域の各側面に重点を置く製品モジュールおよびコンポーネントのスイートが組み込まれたマルチティア・プラットフォームを土台に構築されています。さらに、情報を企業内の配置されている場所で使用するために、その他のサード・パーティー製アプリケーションと統合されます。

Information Server は、ビジネス・インテリジェンス、マスター・データ管理、インフラストラクチャーの合理化、ビジネスの変革、リスクとコンプライアンス (ガバナンス) などの幅広いイニシアチブをサポートします。

ビジネス・インテリジェンス

Information Server では、より優れた意思決定のためのビジネスの統一見解の展開が容易になります。既存のデータ・ソースを理解して、情報のクレンジング、修正、および標準化を行うほか、企業全体で再利用できる分析ビューをロードする上で役立ちます。

マスター・データ管理

Information Server は、さまざまなソース・システムでの情報の保管場所と保管方法を示すことにより、信頼できるマスター・データの作成を簡素化します。また、多種多様なデータを信頼性の高い 1 つのレコードに統合して、情報のクレンジングと標準化を行い、重複を削除して、複数システム間でレコードをリンクします。このマスター・レコードをオペレーショナル・データ・ストア、データウェアハウス、またはマスター・データ・アプリケーションにロードすることができます。また、要求に応じてレコードの全体または一部をアセンブルすることもできます。

インフラストラクチャーの合理化

Information Server は、システム間の関係を示して、インスタンスの統合や旧システムからのデータの移動のためのマイグレーション・ルールを定義することにより、運用コストの削減を支援します。データのクレンジングとマッチングにより、新しいシステムで高品質データを確保できます。

ビジネスの変革

Information Server は、アプリケーション、ビジネス・プロセス、およびポータルに接続できる再利用可能な情報サービスを提供することで、開発を迅速化して、ビジネスの俊敏性を向上させることができます。これらの標準ベースの情報サービスは情報の専門家によって 1 カ所で保守されますが、サービスへのアクセスは企業全体で幅広く可能です。

リスクとコンプライアンス (ガバナンス)

Information Server は、リネージュと品質の証明によって、信頼できる完全な情報のビューを実現することにより、可視性とデータ・ガバナンスの改善を支援します。これらのビューは共有サービスとして幅広く利用可能かつ再利用可能ですが、ビュー内に存在するルールは集中管理されます。

1.2 機能層

InfoSphere Information Server は、図 1-1 に示すように、下記の 3 つの個別コンポーネントを土台に構築された堅固でスケーラブルなサーバー・アーキテクチャーと、シン・クライアントおよびリッチ・クライアントの製品モジュール (クライアント層) で構成されています。

- J2EE アプリケーション・サーバー (サービス層)
- データベース (リポジトリー層)
- 並列処理ランタイム・フレームワーク (エンジン層)

アプリケーション・サーバーとデータベース・サーバーはいずれも標準のサーバー・アプリケーションです。Information Server インフラストラクチャーは介入を最小限に抑えられるように設計されているため、これらのサーバー・アプリケーションを管理するためのスキルは大半の企業が既に持っています。

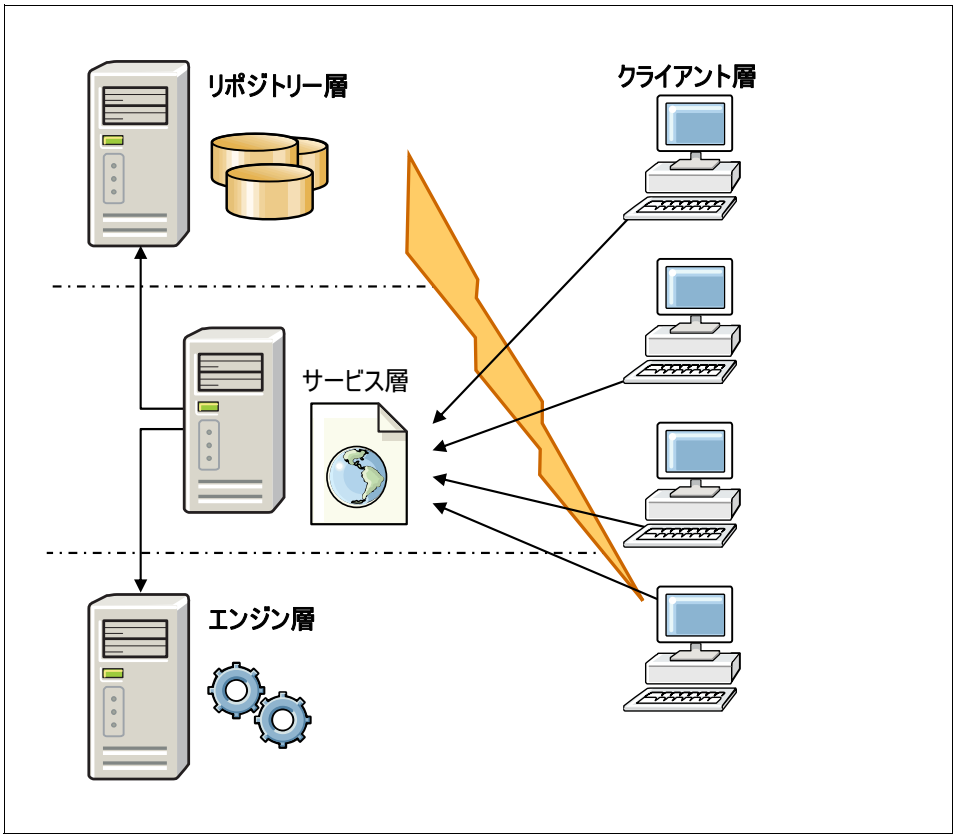


図 1-1　InfoSphere Information Server のアーキテクチャーの概要

Information Server の主な優位性の 1 つは、個々の Information Server 製品モジュールのすべてで共有される、このインフラストラクチャーです。この共有インフラストラクチャーにより、これらのコンポーネントをすべて統合できるため、重複した作業を最小限に抑え、最大限に再利用することができます。さらに、Information Server は、外部アプリケーションからのメタデータをそのリポジトリーに統合して、ビジネス目標を達成するために Information Server 製品モジュールでこの外部メタデータを使用できるようにします。

1.2.1 サービス層

Information Server のサービス層は、IBM WebSphere® Application Server を土台に構築されています。WebSphere Application Server は、認証およびリポジトリー・アクセスなど、すべてのモジュールに共通のサービスのほか、個々の製品モジュールおよびコンポーネント向けの独自のサービスおよび Web アプリケーションのためのインフラストラクチャーを提供します。図 1-2 に、いくつかの製

品モジュール固有のアプリケーションおよびサービスと、すべての製品モジュールで共有されるいくつかの共通サービスを示します。

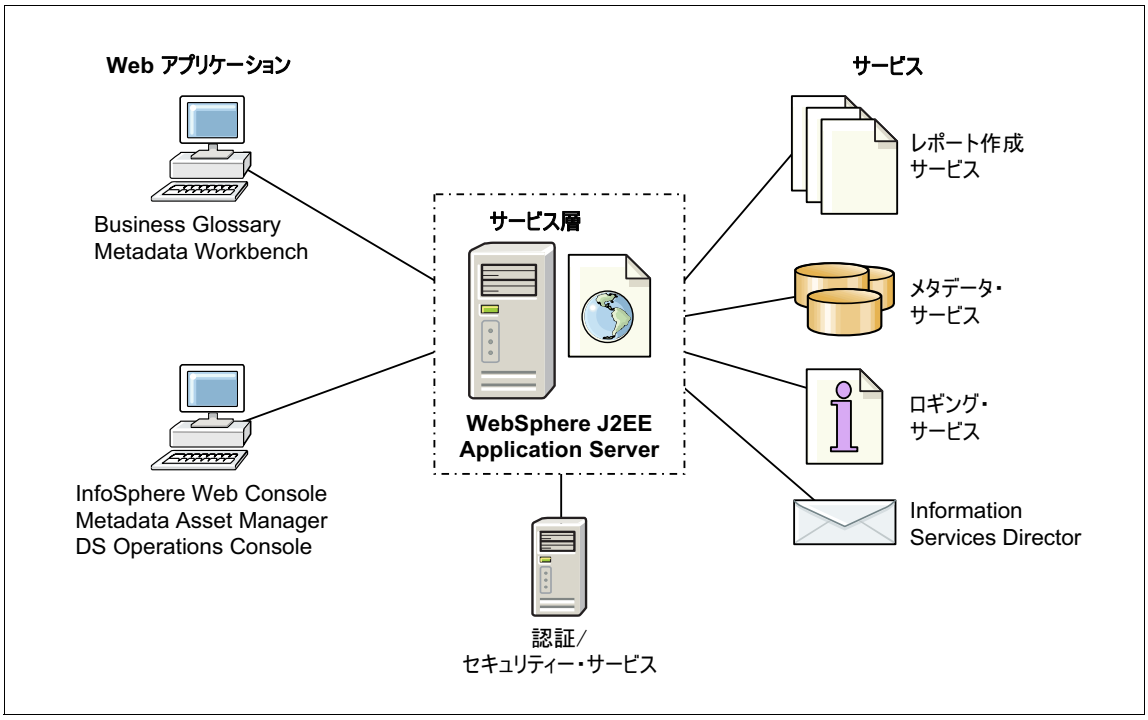


図 1-2　InfoSphere Information Server のサービス層

セキュリティーは、WebSphere Application Server によって管理される主要機能の 1 つです。Information Server が最初にインストールされると、WebSphere Application Server は、自動的に Information Server 内部ユーザー・レジストリー (Information Server リポジトリー内で保持されます) をデフォルトのユーザー・レジストリーとして構成します。その後、Information Server のユーザーおよびグループの管理を簡素化するために、オペレーティング・システムまたは LDAP (例えば、Active Directory) などの別のユーザー・レジストリーを介して認証するように WebSphere Application Server を再構成することが可能です。この領域は、デフォルトの WebSphere Application Server 構成の変更が必要となる可能性がある 2 つの領域のうちの 1 つです。サービス層の負荷が大きくなると、Information Server 専用の WebSphere Application Server Java 仮想マシン (JVM) の最大ヒープ・サイズを増やすことも必要になる可能性があります。

どの認証方式を使用する場合でも、それぞれの製品モジュールおよびコンポーネントは認証のために WebSphere Application Server を使用します。この方法により、Information Server プラットフォーム全体で必要とされ、それぞれの製品モジュールおよびコンポーネントで使用される認証メカニズムは 1 つだけになります。その結果、WebSphere Application Server は、セッションを維持して、アクティブな製品モジュール (セッション) ごとにアクティビティーをログに記録することができます。

各製品モジュールは、それぞれ独自のパーシスタンス層のためにリポジトリーを使用しますが、リポジトリーへの唯一のアクセス方法として WebSphere Application Server リポジトリー・サービスを使用します。WebSphere Application Server は、インストール時に、インストール中に渡される資格情報に基づき、JDBC 層を介してリポジトリーにアクセスするように構成されます。これは、Information Server がデータとメタデータの保全性とセキュリティーを確保する 1 つの方法です。

1.2.2 エンジン層

Information Server のエンジン層は、いくつかの Information Server 製品モジュール向けに並列処理ランタイム機能を提供します。処理能力に対する唯一の制限は使用可能なハードウェアであるため、この機能は、ほぼ無限のスケーラビリティーの基盤となります。重要な点として、並列フレームワークはエンジン層のサーバー・プラットフォーム上で使用可能なすべてのハードウェア・リソースを利用できることに注意してください。そのため、負荷が高い処理のシナリオでは、エンジン層に、他の層と共有していない独自のハードウェア・リソースを割り振ることを強くお勧めします。

図 1-3 (6 ページ) に示すように、エンジン層には、データのクレンジング、プロファイル作成、品質分析、または変換を目的として、より優れたパフォーマンスと機能を利用する、外部データ・ソースへのデータ接続のための組み込みのネイティブ・ドライバーがあります。また、その他多くのデータ・ソースに接続するために、Information Server に組み込まれている ODBC ドライバーやその他の外部ドライバーを使用できます。数百万のレコードの処理が必要となることが多いため、並列処理エンジンは、多種多様なデータにアクセスしてデータを操作するための効率的な方法を提供します。

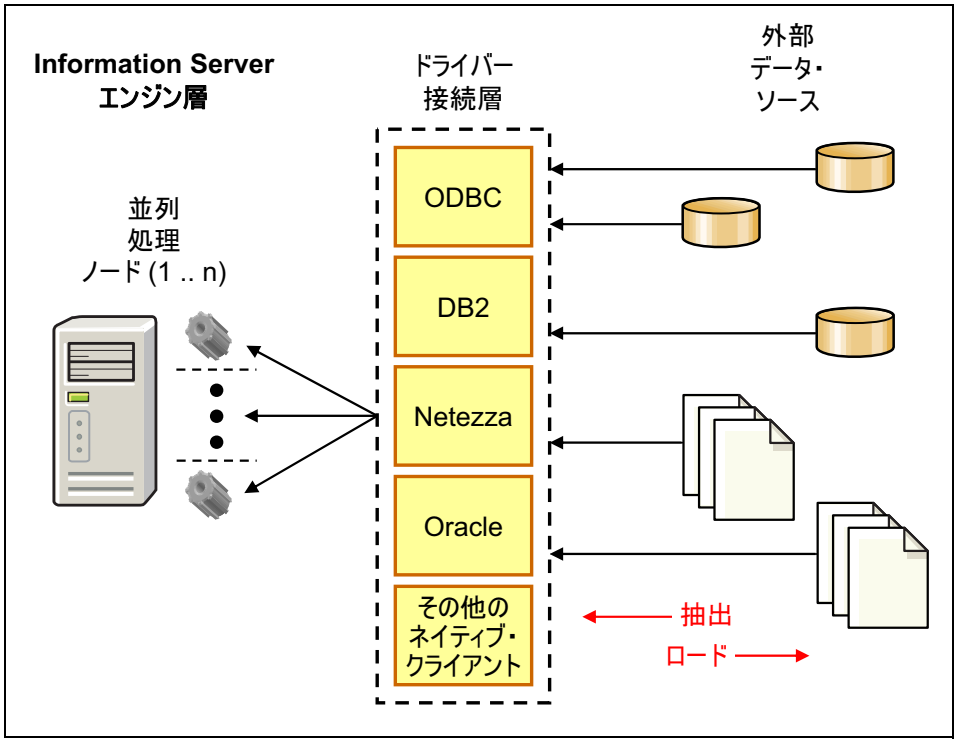


図 1-3　InfoSphere Information Server のエンジン層

1.2.3 リポジトリー層

Information Server のリポジトリー層は、標準の RDBMS (IBM DB2、Oracle、または Microsoft SQL Server) を土台に構築されています。リポジトリー層は、Information Server のすべての製品モジュールおよびコンポーネントのパーシスタンス層を提供します。

(リポジトリー層の中の) Information Server *リポジトリー*は、データ・ソース、データ統合ジョブ、ビジネス・グロッサリー、データウェアハウス、およびデータマートに関する情報と、レポート、およびレポート・モデルがすべて 1 つの共有ロケーションにまとめられている単一リポジトリーです。このリポジトリーは、スイート内の Information Server の製品モジュールおよびコンポーネントで使用できます。

さらに、データ保全性と処理のパフォーマンスを確保して、一時的なパーシスタンスを提供するために、2 つの Information Server 製品モジュール (IBM InfoSphere QualityStage および IBM InfoSphere Information Analyzer) はワークスペースとして独自のスキーマも使用します (図 1-4 (7 ページ) を参照)。処理が実行されると、ワークスペース内の関連するメタデータは、ユーザーが指定する間隔で、Information Server の共有リポジトリーにパブリッシュされ、他の製品モジュールによって使用されるようになります。また、ジョブのモニターとシステム効率の評価に使用されるランタイム、パフォーマンス、およびロギングのデータを収集して保持するために、その他の操作を使用できます。

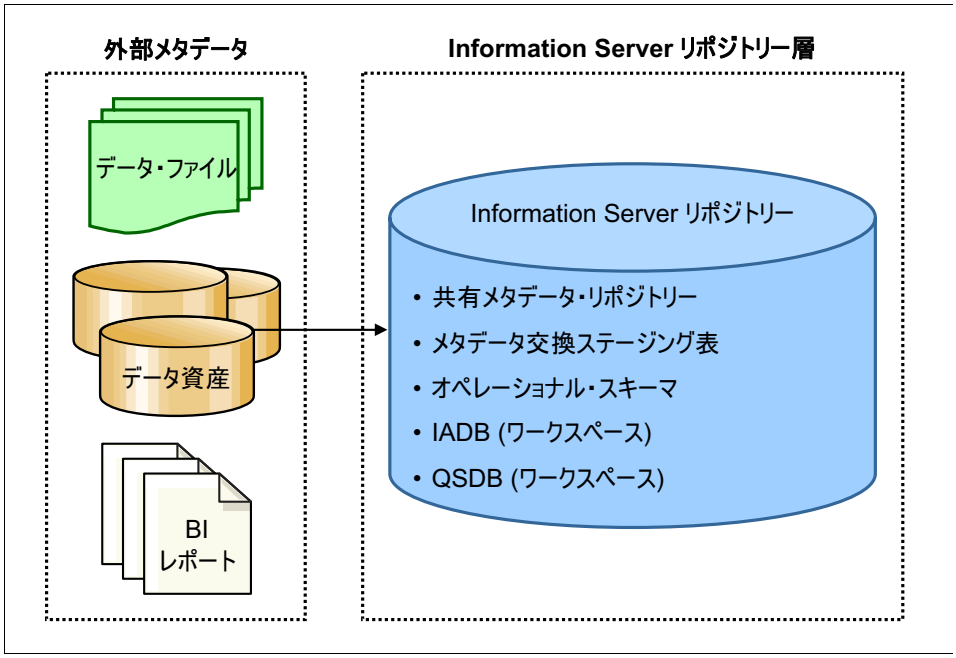


図 1-4　InfoSphere Information Server のリポジトリー層

前述のように、各製品モジュールは、それぞれのデータを Information Server リポジトリーで保持します。したがって、リポジトリー内のメタデータの大部分が内部で生成されたデータです。ただし、1 つ以上の Information Server 製品モジュールまたは機能と関連のある、データ・モデリング・ツール、ビジネス・インテリジェンス・アプリケーション、その他の構造化データのソースなどの外部アプリケーションからメタデータをインポート、ロード、および付加することも可能です。Information Server は、ユーザーがこのサード・パーティーのメタデータをステージング領域にインポートするための機能を備えています。このステージング領域は、外部メタデータ用の一時的なロード領域を提供します。この領域では、この新しいメタデータを (ロード前に) 既存のメタデータと比較して、その影響を評価して処理するほか、ロードされたメタデータのレコードを維持することもできます。

外部アプリケーションがメタデータへの (直接的または抽出による) アクセスを提供しない場合は、制御された手動プロセスとツールによってこれらの関連する外部情報資産をリポジトリーで文書化することも可能です。

1.3 製品モジュール

InfoSphere Information Server は、情報統合ドメイン内でビジネス目標を達成するために連携する製品モジュールとコンポーネントの集合を提供します。製品モジュールは、計画から設計、実装、およびレポーティングの段階に至るイニシアチブ全体を通じてビジネスと技術の機能を提供するよう設計されています。

Information Server は、IBM InfoSphere Blueprint Director、IBM InfoSphere DataStage、IBM InfoSphere QualityStage、IBM InfoSphere Information Analyzer、IBM InfoSphere FastTrack、IBM InfoSphere Business Glossary、および IBM InfoSphere Metadata Workbench のほか、IBM InfoSphere Data Architect および IBM InfoSphere Discovery の製品モジュールおよびコンポーネントで構成されています。これらは一体的に関連したアプリケーションです。製品モジュールに加えて、Information Server には、外部資産をリポジトリーにインポートして、1 つの環境から別の環境に内部資産をデプロイし、これらの資産を管理して重複とオーファンを削除するための以下のユーティリティーが組み込まれています。

- InfoSphere Information Server Manager
- istool コマンド・ライン・ユーティリティー (CLI)
- IBM InfoSphere Metadata Asset Manager

このセクションでは、これらの製品モジュールとユーティリティーのそれぞれについて、機能の概要、関連付けられているメタデータ、および基盤とするインフラストラクチャー・コンポーネントを説明します。

1.3.1 InfoSphere Blueprint Director

InfoSphere Blueprint Director は主に、情報のガバナンス、情報の統合、データ品質、ビジネス・インテリジェンス、またはその他の種類の情報ベースのプロジェクトにおいて InfoSphere Information Server ベースのイニシアチブの計画の概要を作成するために使用されるグラフィカル設計ツールです。タスクを簡素化するために、Blueprint Director は、プロジェクトに適合するように簡単にカスタマイズ可能ですぐに使用できるいくつかのプロジェクト・タイプ・ベースのコンテンツ・テンプレートとバンドルされています。あるいは、必要に応じて新しいブループリントを作成することもできます。

Blueprint Director は、プロセスやタスクなどを表す標準グラフィカル・オブジェクトがドラッグ・アンド・ドロップされる設計キャンバスを備えています。オブジェクトを相互に接続できるため、これらのイベントの順序またはイベント間の依存関係を暗黙に示すことができます。各グラフィカル・オブジェクトにはその目的を示すラベルが付いていますが、オプションで IBM Rational® Method Composer で作成およびパブリッシュされたコンテンツにオブジェクトをリンクして、さらに詳細な情報を確認することができます。単一オブジェクトが多数のタスクまたはプロセスを表す場合、オブジェクトをドリルダウンして、ブループリント・ダイアグラムのさらに詳細な層を作成したり、詳細な層にリンクし、最終的なブループリントにプロセス (およびサブプロセス) の複数の階層レベルが含まれるようにすることができます。メソッド (テキスト記述) と結合された階層的なブループリント・ダイアグラムは、上から下まで (高水準から低水準まで) 構築されたプロジェクト計画の基盤を形成します。

Blueprint Director は、パーシスタンス、認証、またはその他の共有サービスのために Information Server インフラストラクチャーに依存しないスタンドアロンで Eclipse ベースのクライアント専用アプリケーションという点で、Information Server 製品モジュールおよびコンポーネントの中でも独特のコンポーネントです。そのため、インフラストラクチャー全体が使用可能になる前の早期の段階

でプロジェクトを計画できる有益な柔軟性を得られます。ただし、Information Server (IS) バージョン 9.1 では、Metadata Workbench (MWB) または Business Glossary (BG) を介して表示できるブループリント (読み取り専用) を Information Server にパブリッシュできるようになる予定です。

本書で基準として取り上げる Blueprint Director V8.5 は現時点ではいずれかの Information Server 製品モジュールおよびコンポーネントによって利用されるメタデータを生成しませんが、新しい V9.1 は生成します。ただし、V8.5 には、XML 形式 (*.bpd) のファイルの独自のパーシスタンス層があります。また、V8.5 には、Information Server リポジトリー・オブジェクトにリンクして、さらには Information Server 製品モジュールおよびコンポーネントを起動し、これらのリンクされたオブジェクトをネイティブ・ツールで表示する機構があります。

1.3.2 InfoSphere DataStage および InfoSphere QualityStage

InfoSphere DataStage および InfoSphere QualityStage は、幅広いプロジェクトに不可欠な統合機能を提供します。DataStage は、複数のソースおよびターゲットのアプリケーションにわたってデータを統合し、大量のデータを収集、変換および配信します。QualityStage は、マスター・データの標準化から非重複化、確立に至るデータ・クレンジング機能を提供します。どちらの製品モジュールも、(エンジン層の) 並列処理ランタイム・エンジンを活用して、データ・フローと使用可能なシステム・リソースを最適化するスケーラブルな方法でジョブを効率的に実行できます。

一般的に、DataStage と QualityStage の両方のジョブは (DataStage および QualityStage Director、コマンド・ライン、またはエンタープライズ・スケジューラーなどを使用して) バッチ操作として起動されます。ただし、並列処理ランタイム・エンジンを使用しながら、IBM InfoSphere Services Director (ISD) を使用してこれらのジョブを持続的にサービスとして (リアルタイムで) 実行することができます。「常時実行」シナリオでは、ジョブをデプロイするために、WebSphere Application Server サービス・バスの ISD を使用して、さらに Information Server のサービス層のメリットを使用します。

DataStage および QualityStage は、DataStage and QualityStage Designer という豊富な機能を備えた同じユーザー・インターフェースを共有します。このインターフェースは、次のように、抽出・変換・ロード (ETL) を実行するジョブを設計するための簡単な方法を提供します。

- 多様なデータ・ソースからデータを*抽出*します
- ビジネス要件に従ってデータを*変換* (または処理) します
- 結果として生成されるデータを関連するターゲット・データ・ストレージ・システムに*ロード*します

DataStage および QualityStage は、プロジェクトを管理して、ジョブのテスト、実行、およびトラブルシューティングを行うために、次の追加のリッチ・クライアント・アプリケーションも共有します。

- DataStage and QualityStage Administrator
- DataStage and QualityStage Director

また、サービス層の DataStage and QualityStage Operations Console という Web アプリケーションを、追加のランタイム・ジョブおよびシステム・モニターとパフォーマンスのメトリックを提供するエンジン層のエージェントと共有します。

DataStage および QualityStage のプロジェクトは Information Server リポジトリーで保持されます。リポジトリー・ストレージ・モデルのこの部分には、DataStage および QualityStage のジョブデザインと、表定義、コンテナー、パラメーター・セット、コネクター・オブジェクトなどのその他のプロジェクト成果物が入っています。DataStage および QualityStage の設計と、オプションのランタイム・オペレーショナル・メタデータは、リポジトリー内の累積メタデータに基づいて影響とリネージュのレポートを計算するためのキー・コンポーネ

ントです。設計とオペレーショナル・メタデータを生成するほか、DataStage および QualityStage のジョブは、ジョブ内の DataStage および QualityStage のステージ・リンクの設計で使用される表定義を取り込むために物理メタデータを利用します。

1.3.3 InfoSphere Information Analyzer

IBM InfoSphere Information Analyzer (IA®) は、データ品質の評価とモニター、データ品質に関する問題の特定、コンプライアンスの実証、監査証跡の維持に役立ちます。IA Workbench は、(データ・ルールによる) 品質評価と (列分析による) データのプロファイル作成という 2 つの機能を実行するグラフィカル・ユーザー・インターフェースを提供するリッチ・クライアントです。Information Analyzer では、これらのアクティビティーを実行する前に、接続情報 (ソースへの接続に使用されるドライバーや、DSN および URL など) を含めて、これらのデータ・ソースからのメタデータがリポジトリー内に存在することが必要です。このテクニカル・メタデータのインポートは、InfoSphere Information Analyzer、または InfoSphere Metadata Asset Manager という Information Server ユーティリティーから実行できます (1.3.9『Information Server Manager、ISTOOL、および InfoSphere Metadata Asset Manager』（14 ページ）を参照)。

Information Analyzer は、エンジン層で並列エンジンの接続と並列処理の機能を使用してデータ・ソースを照会し、リポジトリー層で列値を Information Analyzer データベース・ワークスペース (IADB) にロードします。Information Analyzer は、次に、選択された各列のプロファイル分析を個別に実行します。すべての分析結果は、プロセスの再実行によってリフレッシュされるまで、IADB ワークスペースで保持されますが、比較のためにベースラインを設定することもできます。すべてのプロセスは、WebSphere Application Server のサービス層で実行されるプロプラエタリー・サービスを通じて調整されます。処理が完了すると、分析結果を IADB ワークスペースから共有リポジトリーにパブリッシュできます。そこで、その他の製品モジュールおよびコンポーネント (特に DataStage および FastTrack) で分析結果を使用できます。

Information Analyzer はデータ品質のレベルを評価するために、データが順守すべき (あるいは、順守すべきでない) 制約 (データ・ルール) を設定して、データがこれらのルールに準拠しているかどうかをテストします。この機能は、データ品質と情報ガバナンスのイニシアチブの一部として行うことが多い持続的なデータのモニター (トレンド分析) を実行します。

1.3.4 InfoSphere FastTrack

InfoSphere FastTrack は、データ統合ジョブの開発者が使用するソースからターゲットへの (または技術的にはターゲットからソースへの) マッピング仕様を作成するよう設計されたリッチ・クライアントです。クライアント・アプリケーションのメインパネルは、オブジェクトをリポジトリーのディスプレイからソース列、ターゲット列、および割り当てられたビジネス用語のセルに入力 (コピーまたはドラッグ) するためのスプレッドシートのような縦欄構造を提供します。また、ソースからターゲットへの変換と、オプションで DataStage Transformer ステージ固有のコードを手動で記述することもできます。

完成したマッピング仕様は、FastTrack によって直接的に生成される注釈付きの DataStage ジョブなど、いくつかの形式で出力することができます。この形式は、特に DataStage 開発者にとって有用です。開発者が使い慣れた方法で仕様が配信されるためです。さらに、この配信形式は、新しいジョブにそのままの状態でコピーできる設計成果物を含めて、新しいジョブを作成するための基盤として使用できるジョブ・テンプレートを提供します。

マッピング仕様文書のために FastTrack を使用するその他の利点として、以下の機能が挙げられます。

- 仕様を中央の場所で保管および管理

- ソースとターゲットの列を指定するためのシンプルなドラッグ・アンド・ドロップ機能
- 正しいスペリングが保証された (実際にリポジトリー内に存在する) ソースとターゲットの列名の正確性
- ソースとターゲットの各列のデータ・タイプなどのメタデータの可視性。適切なマッピング関係と変換を判別する上で役立ちます。
- パブリッシュされたデータ・プロファイルの結果、名前認識、およびビジネス用語の割り当てに基づくマッピング、結合、およびルックアップの支援の「発見」
- リネージュ・レポートで使用可能な共有リポジトリー内のパーシスタンス層

1.3.5 InfoSphere Business Glossary

InfoSphere Business Glossary は、主にシン・クライアント・ブラウザー・アプローチを使用しますが、組織全体でユーザーが情報を共有できるようにその他のインターフェースもいくつか使用します。基本的なコンテンツは、内容、参照、およびコンテキストを示すカテゴリーごとに用語の定義が編成されている用語のグロッサリーです。用語とカテゴリーのどちらにも、いくつかの記述属性と、他の (技術関連とビジネス関連の両方の) Information Server リポジトリー・オブジェクトとの関係、情報スチュワード、開発サイクルの状況を定義するその他の属性や、オプションのユーザーが作成したカスタム属性が含まれています。

用語がグロッサリーにパブリッシュされた後、ユーザーは、以下のインターフェースで使用できる多様な検索および表示機能により、それらにアクセスできます。

- Business Glossary (Web ブラウザー・クライアント)
- Business Glossary Anywhere ポップアップ・クライアント (ワークステーション上のリッチ・クライアント)
- IBM Cognos® のコンテキストに依存したグロッサリーの検索および表示 (メニュー・オプション)
- InfoSphere Data Architect の Eclipse プラグイン (REST API を使用) およびその他の Eclipse ベースの製品
- カスタム・プログラミングによる REST API のプログラマブル・アクセス

Business Glossary 関連のインターフェースで使用できる検索と表示の機能に加えて、Information Analyzer および FastTrack は、ビジネス・グロッサリーに含まれているビジネス・メタデータの明確なビューを提供します。InfoSphere Metadata Workbench もビジネス・メタデータの明確なビューを提供しますが、Business Glossary からビジネス・メタデータを確実に組み込むことができるカスタム・メタデータ照会も提供します。

Business Glossary ブラウザーは、グロッサリーのオーサリング用の唯一のグラフィカル・ユーザー・インターフェース (用語とカテゴリーの作成、変更、および削除のための読み取り / 書き込みインターフェース) を提供すると同時に、検索と表示のためのメインツール (読み取り専用) でもあります。使用可能な機能は、Information Server 管理者によって定義されるロールに基づいています。ただし、さらにきめ細かい追加のアクセス制御が存在します。この制御は、Business Glossary 管理者のロールを持つすべてのユーザーによって管理されます。

Business Glossary は、ビジネス・グロッサリーを作成および変更するための機能的な Web ブラウザー・インターフェースを提供します。このインターフェースでは、管理者は、適切にフォーマットされた CSV および XML ファイルから既存のグロッサリー情報を一括ロードすることができます。この方法は、外部のアプリケーションやスプレッドシートに含まれている可能性があるグロッサ

リー情報を使用するための有用な機構です。さらに、既存のコンテンツを取得したり、コンテンツを変更したりするためにカスタム・アプリケーション開発を通じてプログラムでアクセスできる API (REST) があります。

Business Glossary は、すべてのデータ (ビジネス・メタデータ) を Information Server リポジトリーで保持します。ビジネス用語には既にリポジトリー内に存在する物理資産 (数例を挙げると表、列、レポートのヘッダー) との関係がある可能性があるため、このアプローチは特に適しています。さらに、グロッサリーの作成や変更のほか、Business Glossary を使用して、パブリッシュされたビジネス・メタデータの表示とリポジトリー内のその他のメタデータ・オブジェクトの表示の両方が可能です。これは、InfoSphere Metadata Workbench と似ていますが、別のビューのパースペクティブから表示する方法です。

1.3.6 InfoSphere Metadata Workbench

InfoSphere Metadata Workbench の主な機能は、Information Server リポジトリーのコンテンツの「ビュー」を提供することです。そのため、ユーザーは、特定の資産の発信元、履歴、および所有権に関する質問への回答を得ることができます。この「ビュー」は、次のようにいくつかの形式で使用できます。

- 事前構成された 3 つの Metadata Workbench レポート : 影響分析、ビジネス・リネージュ (Business Glossary からも使用可能)、およびデータ・リネージュ。メタデータに基づくこれらのレポートは、メタデータ・オブジェクト間の関係と接続のほか、メタデータ・チェーン内のオブジェクトへの変更による潜在的な影響を重点的に示します。
- シンプルな検索と、複雑な「随時」照会および永続化されたカスタム照会。この場合、ユーザーは、一連の制約に対して、表示するメタデータ属性のリストを指定します。
- メタデータ資産のカテゴリー (ホスト、表、列、用語、BI レポート、およびその他のメタデータ・オブジェクトなど) を表示できる表示機能。これにより、ユーザーは、「ドリルダウン」表示の対象としてメタデータの特定のインスタンスを選択できます。

これらのレポートはいずれも画面で表示されますが、外部での使用やさらなる分析のために CSV 形式のファイルにエクスポートすることもできます。

リポジトリーのコンテンツのビューおよびレポートのほか、Metadata Workbench は、説明とカスタム属性や、その他のデータベース・メタデータ、ビジネス用語、モデル・オブジェクト、さらにはデータ・スチュワード (つまり、読み取り / 書き込み機能) などの関連メタデータ資産を割り当てることにより、既存のメタデータを拡張できます。さらに、Metadata Workbench は、Information Server 製品モジュールおよびコンポーネントによって自動的に生成されないメタデータや、外部アプリケーションから容易にインポートできないメタデータをリポジトリーに追加することを目的として、拡張リネージュ・オブジェクトを「作成」できます。このアプローチは、外部プロセスの文書化と、Information Server のクライアントごとの使用法の境界を超えたビジネスまたはデータのリネージュ・レポートの拡張の両方を実現します。

ワークベンチからデータ・リネージュ・レポートを表示したり、照会 (随時レポート) を呼び出したりするには、特に数百の情報資産について分析とレポーティングを行う場合は、Metadata Workbench 内でさらに多くのシステム・リソースと構成が必要になる場合があります。そのようなアクションでは、Information Server のコンポーネント間で共有されているサービス層の使用可能なリソースを使用します。Metadata Workbench は、設計上、サービス層の過負荷を防止するために、システム構成に従ってこのようなアクションの結果を制限します。したがって、サービス層の実際のデプロイ済み構成に合致するように、システム構成を拡張することが必要になる可能性があります。

1.3.7 InfoSphere Discovery

InfoSphere Discovery は、データ関係を自動的に発見するソリューションです。これにより、データのコンテンツ、関係、および変換を理解して、ビジネス・オブジェクトを発見し、データ・ストア内および複数の異種データ・ストア間で機密データを識別することができます。Discovery から得られる自動化された結果は、特に現在多くの組織が用いている手動の (自動化されていない) データ分析アプローチと比較すると、すぐに使用可能で、正確で、簡単に生成することができます。

Discovery は、*自動的に*データ・ソースを分析してデータに関する仮説を生成することによって機能します。プロセス全体を通じて、データを調査し、*データ・プロファイル*または*列分析*を含むメタデータを生成します。

情報統合プロジェクトのコンテキストでは、このメタデータにより、データとそれらの関係を理解できるようになります。このメタデータをガバナンスのために使用したり、ETL 仕様の計画支援として簡単にソースからターゲットへのマッピングに役立てたりすることができます。

Discovery は、データ関係を発見する目的で、直接、ソース・データ・システムにアクセスしてデータを取得します。通常は Business Glossary、FastTrack、または両方のために結果を Information Server リポジトリーで共有する準備ができると、エクスポートとインポートのユーティリティーは、Information Server の共有リポジトリーに結果をパブリッシュできます。これらの結果は、物理メタデータを拡張して、ビジネス用語を物理資産と関連付け、また、多種多様なデータ・ストアにわたるデータのコンテンツと関係に基づいて ETL のマッピング仕様の設計に役立ちます。

1.3.8 InfoSphere Data Architect

InfoSphere Data Architect は、企業向けのデータ・モデリングおよび統合設計ツールです。このツールを使用して、分散されている多様なデータ資産について発見、モデル化、可視化、関連付け、および標準化を行うことができます。スタンドアロンで Eclipse ベースのクライアント専用製品モジュールであり、独自のパーシスタンス層 (XML 形式のファイル *.dbm、*.ldm、*.ndm、*.ddm) を持ち、多様なモデル間のクロスファイル・リンク (関係) を持つ可能性がある点で、Blueprint Director と構造的に似ています。

トップダウン・アプローチで Data Architect を使用して、論理モデルを設計し、論理ソースから物理データ・モデルを自動的に生成することができます。DDL スクリプトを物理データ・モデルから生成して、その設計に基づいてデータベース・スキーマを作成することができます。Data Architect が持つもう 1 つの可能性は、RDBMS に接続して、Data Architect の物理データ・モデルから直接的にデータベース・スキーマをインスタンス化することです。この「生成」機能は両方向で作動するため、変更、再利用、バージョン管理、およびその他の目的で、既存のインスタンス化されたデータベースを Data Architect の物理データ・モデルに「リバース・エンジニアリング」することも可能です。

モデルを最初から設計するのではなく、Data Architect で利用できるフォーマットの IBM Industry Models を購入するアプローチが考えられます。この方法では、プロジェクトのデータベース設計段階をすぐに開始でき、特定の業界における IBM のデータ・モデリングに関する豊富な専門知識を活用できます。標準的な手法では、業界標準の論理モデルのスコープをお客様の要件にも適合するように設定して、業界標準とお客様の詳細を組み合わせた適切な物理データ・モデルを作成します。InfoSphere Data Architect 向けの IBM Industry Models パッケージに加わった利点は、Data Architect の論理モデルとの関係 (割り当て済み資産) を備えた業界標準グロッサリー・モデルが組み込まれていることです。これを使用して、Business Glossary にデータを取り込むことができます。

Data Architect は、共有サービスやパーシスタンスのために Information Server インフラストラクチャーに依存しません。ただし、InfoSphere Metadata Asset Manager ユーティリティー (1.3.9『Information Server Manager、ISTOOL、および InfoSphere Metadata Asset Manager』（14 ページ）を参照) を使用して、Data Architect モデル (論理、データ、およびグロッサリー) を Information Server リポジトリーにインポートすることができます。さらに、Business Glossary Eclipse (BGE) plug-in for Data Architect を使用して、「ドラッグ・アンド・ドロップ」機能により、Business Glossary の用語とカテゴリーを Data Architect の論理モデル・エンティティーおよび属性と、データ・モデルの表および列に直接関連付けることもできます。BGE により、Business Glossary は、Data Architect で使用できるようにオフラインの XML 形式のファイルでダウンロードされます。このオフラインのグロッサリーは、必要に応じて手動で Business Glossary と同期できます。ただし、これらの 2 つのグロッサリーが同期されていない場合は、Data Architect が起動されるたびに通知されます。

1.3.9 Information Server Manager、ISTOOL、および InfoSphere Metadata Asset Manager

それぞれの Information Server 製品モジュールおよびコンポーネントは、Information Server リポジトリーとの間でメタデータを生成して利用するため、リポジトリーを管理するためのメカニズムを準備することが必要です。そのため、Information Server には、Information Server メタデータ・リポジトリーのさまざまな側面を管理するための次の 3 つの主要なユーティリティーがあります。

- Information Server Manager
- ISTOOL
- InfoSphere Metadata Asset Manager

Information Server Manager Console は、リッチ・クライアント・ユーザー・インターフェースです。Information Server の 1 つ以上のインスタンスに接続して、管理者が 1 つ以上の DataStage または QualityStage プロジェクトからの DataStage または QualityStage オブジェクト (およびオプションでそれらの従属オブジェクト) をパッケージに編成できるようにします。これらのパッケージは、バージョン管理 (および後続のデプロイメントとインポート) のためのファイルとしてエクスポートしたり、Information Server の別のインスタンスに直接的にデプロイしたり (例えば、開発からテスト、テストから疑似本番、またはお客様のデプロイメント・アーキテクチャーに適合するその他のプロモーションのシナリオ) することができます。

Information Server Manager Console を使用してパッケージが *定義* された後、パッケージ・ファイルのファイル作成 (エクスポート) およびデプロイメント (インポート) は、**istool** というコマンド・ライン・ユーティリティー (CLI) によっても 実行することができます。**istool** CLI は、クライアント・ワークステーションと Information Server ホストの両方にインストールされます。管理者が対話式に開始するか、スクリプト記述して使用法を標準化することができます。一般的な方法では、プロセスを自動化してセキュリティーを最大限に強化するために、**istool** CLI のファイル作成およびデプロイメントのスクリプトはエンタープライズ・スケジューラーによって実行されます。

DataStage および QualityStage のパッケージ・ファイル作成およびデプロイメントの機能に加えて、**istool** CLI は、その他すべての Information Server 製品モジュールおよびコンポーネントから作成される次のアイテムのようなメタデータを .ISX アーカイブ・ファイルにエクスポートするために使用されます。

- Business Glossary: 用語およびカテゴリー
- FastTrack: プロジェクトおよびマッピング仕様オブジェクト
- Information Analyzer: プロジェクトおよびルール
- 共有 (共通) リポジトリー : 物理データ・リソース (共通メタデータ)

これらのアーカイブ・ファイルは、その後、DataStage および QualityStage パッケージ・ファイルと同様に、開発サイクルのデプロイメント (開発 - テスト - 本

番) や、より新しいバージョンの Information Server へのマイグレーションのために、Information Server の他のインスタンスにインポートできます。

InfoSphere Metadata Asset Manager は、Information Server スイートに次のようないくつかの機能を提供する Web ベース・コンポーネントです。

- 外部ソース (RDBMS、ビジネス・インテリジェンス、モデリング・ツール、データ・ファイル) からメタデータをステージング領域にインポートして、手動での競合解決を目的として既存のメタデータと比較します。
- 承認されたメタデータをステージング領域からリポジトリーにロードします。
- リポジトリー内の重複したメタデータを管理 (マージおよび削除) します。

InfoSphere Metadata Asset Manager は、Metadata Integration Bridges を使用します。これは、外部ソースからのメタデータを、Information Server にロードして使用できるフォーマットに変換します。ただし、DataStage と QualityStage、Information Analyzer、および FastTrack で使用されているのと同じ「コネクター」機能も使用して、互換性のある RDBMS および ODBC データ・ソースに直接接続します。また、Import Export Manager の大半の機能を、前述の追加機能を提供する拡張インターフェースに置き換えます。

2

Information Server のインストール・トポロジー

本章では、基本的な 1 台のコンピューターから可用性の高いクラスターおよびグリッド構成に及ぶさまざまな IBM InfoSphere Information Server (Information Server) のインストール・トポロジーについて説明します。

この章では、Information Server の主要なソフトウェアのサブシステム・コンポーネント (層) のそれぞれについて、組み合わせおよび単独でのデプロイメント・アーキテクチャーを考察します。ただし、重点は、単一のインストール環境全体 (例えば、開発、テスト、本番) に制約されます。複数の環境に関する考慮事項については後続の章で説明します。本章では、以下のトポロジーおよび関連トピックについて説明します。

- 単一コンピューター
- クライアント / サーバー
- 層ごとに専用のコンピューター
- 専用エンジン
- クラスター
- アクティブ / パッシブ
- アクティブ / アクティブ
- 超並列処理
- グリッド
- Web サービス
- 災害復旧

2.1 Information Server のソフトウェア層

図 2-1 (18 ページ) に示すように、Information Server には 4 つの層 (1 つのクライアント層と 3 つのサーバー層) があります。

- クライアント層 : このクライアント層は、Windows インターフェースを介してユーザーに製品インターフェースを提示します。インストールされているクライアント・コンポーネントは、Java および .Net のコンポーネントで構成されています。
- サービス層 : このサーバー層は、WebSphere Application Server を使用する共通のサービスと製品固有のサービスを配信します。サービス層の中では Business Glossary および Metadata Workbench の製品ソフトウェアが主にホストされており、残りの製品モジュールのコンポーネントと製品モジュールで共通のサービスもホストされています。
- エンジン層 : このサーバー層は、DataStage、QualityStage、および Information Analyzer の並列エンジンとフレームワークを提供します。また、データベース・コネクター、パック、サービス・エージェント、および DataStage サーバー・コンポーネントも格納しています。
- Information Server リポジトリー層 : このサーバー層は、データベースを使用して実装されるメタデータ・パーシスタンス層を提供します。データベースには、DataStage ジョブデザインおよび仕様など、ビジネス・メタデータ、オペレーショナル・メタデータ、およびテクニカル・メタデータが格納されています。

図 2-1 は、単一リポジトリーを使用するようにクライアント層、サービス層、およびエンジン層の 1 から N のインスタンスを構成できることも示しています。

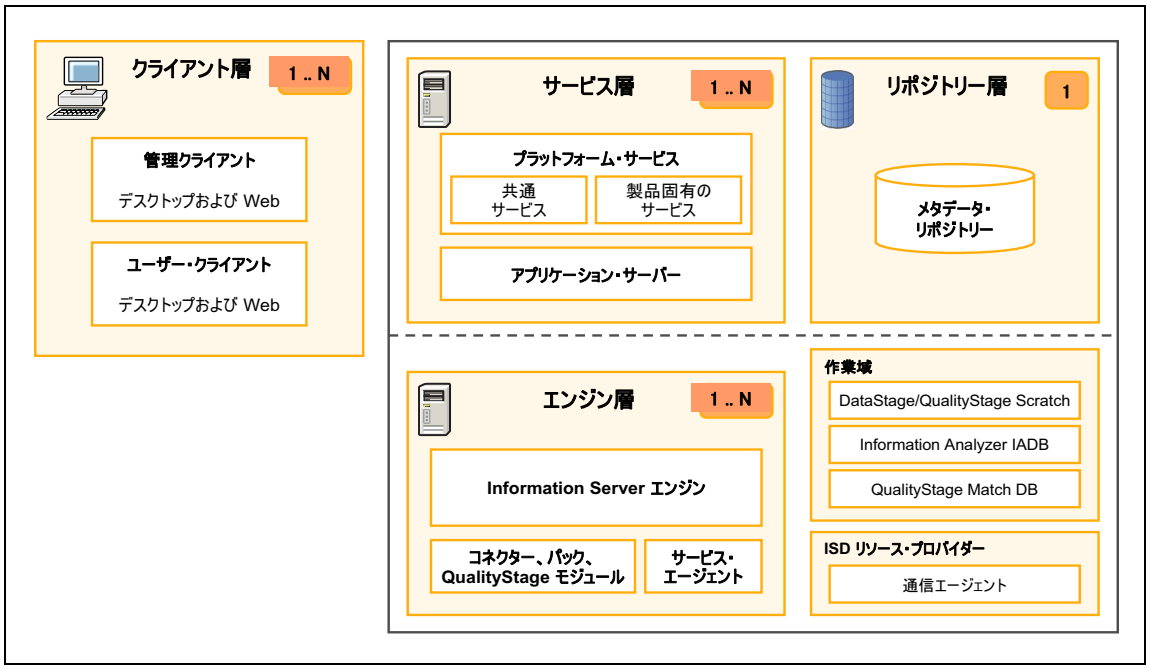


図 2-1 Information Server のソフトウェア層

2.2 サポートされるプラットフォーム

Information Server の各層は、以下のプラットフォームで使用可能です。

- ユーザー・インターフェースを提供する、インストール可能なクライアント層のコンポーネントは、Windows プラットフォームでのみ使用できます。Business Glossary および Metadata Workbench で必要になるのは、サポートされる Web ブラウザーだけです。これらの 2 つの Information Server モジュールには、クライアントのインストール可能なコンポーネントはありません。
- サーバー層 (サービス、エンジン、リポジトリー) は、UNIX、Linux、および Windows プラットフォーム (Microsoft Windows Server、Red Hat Linux、SUSE Linux、IBM AIX®、Oracle Sun Solaris、および Hewlett Packard HP-UX) で使用可能です。各サービス層コンポーネントを別々のホストにデプロイすることはできますが、サービス層とエンジン層は同じタイプのプラットフォームにデプロイする必要があります。
- Information Server リポジトリーのデータベースは、DB2、Oracle、または SQL サーバーを使用して実装できます。

デプロイメント・トポロジーで、(例えば、長距離が原因で) 帯域幅、遅延、または両方の点で優れていない通信リンクを介してクライアントがサーバー層に接続される場合、リモート・デスクトップ (例えば、クライアント・アクセス・アーキテクチャーの端末サービスまたは Citrix インストール済み環境) の使用を検討してください。

InfoSphere Information Server のシステム要件は、システムの範囲と規模に応じて異なります。十分なパフォーマンスを発揮する環境をサポートするのに必要となる正確な構成は、システムまたは環境の可用性に関する要件のほか、サーバーの速度、メモリー、ディスク入出力、データ・ボリューム、ネットワークとサーバーのワークロードといった多数の要因によって異なります。(本書ではキャパシティー・プランニングについては説明していません。ただし、具体的な実装の規模を決定する際、IBM の技術専門家または IBM Techline に支援を要請できます。)

本章の残りの部分では、単一環境をインストールするためのさまざまなインストール・トポロジーと、それぞれのメリットと考慮事項について説明します。考えられるすべての層とクラスターの技術的な組み合わせについては説明していませんが、本章では、どのような状況でも十分な情報に基づいて選択できるように、該当する組み合わせと原則について説明しています。

2.3 単一コンピューター・トポロジー

図 2-2 に示すように、この構成では、4 つのすべての層が同じコンピューター上にあります。このトポロジーは、デモンストレーション・システムや小規模の開発環境に適しています。

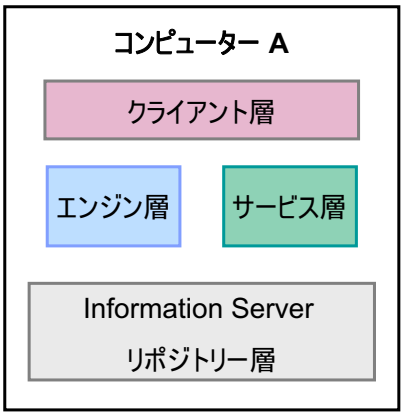


図 2-2 単一コンピューター・インストール

このトポロジーのメリットと考慮事項

メリットを下記に示します。

- ハードウェアのデプロイメントがシンプルです。
- インストールがシンプルです。
- サーバー間ネットワークまたはセキュリティー構成がありません。
- 適切なサイズのノートブックのインストールが可能です。

考慮事項を下記に示します。

- クライアント要件のために、この構成では本質的に、すべての層がサポートされる Microsoft Windows プラットフォーム上に実装される必要があります。この考慮事項は、UNIX または Linux サーバー環境のみを主に使用する組織にとって懸案事項となる可能性があります。
- 4 つの層のすべてが同じマシン・リソースを獲得しようと競合するため、スケーラビリティーは考慮すべき事項です。
- ハードウェア障害が起こると、サービスの完全な損失につながる可能性があります。
- 複数のユーザーに端末サービスが必要です。

2.4 クライアント / サーバー・インストール・トポロジー

このトポロジーでは、サービス層、エンジン層、および Information Server リポジトリー層はすべて 1 台のコンピューターにインストールされます。クライアント層は、別のコンピューターにインストールされます。

インストール可能なクライアント・モジュールをホストするクライアント層のコンピューターは Microsoft Windows を実行する必要があります。その他の層をホストするコンピューターは、IBM InfoSphere Information Server がサポートする任意のオペレーティング・システムを実行できます。

図 2-3 に示されているこのトポロジーは、管理を一元化して、管理をクライアント・ユーザーから分離します。各ユーザーがクライアント層コンピューターを使用する可能性があります。

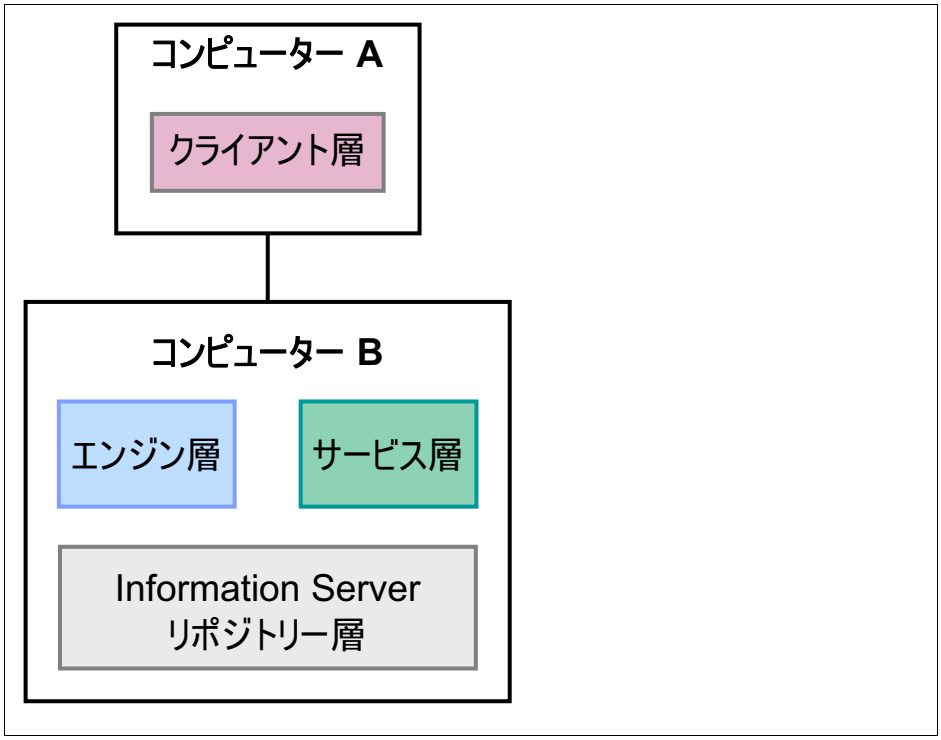


図 2-3　クライアント / サーバー・トポロジー

このトポロジーのメリットと考慮事項

メリットを下記に示します。

- サーバー稼働環境のハードウェアの完全に柔軟な選択肢は、Information Server でサポートされるプラットフォームと一致します。ご使用のハードウェア・プラットフォームが Information Server でサポートされていれば、サーバー稼働環境の選択において完全な柔軟性を得られます。
- クライアントとサーバー・コンポーネントの管理が、より明確に分離されます。
- 高可用性 (HA) とフェイルオーバーは、極めてシンプルで、共有ファイル・システムまたは SAN 全体でサーバーを基盤として行われます。
- 特に 2.3『単一コンピューター・トポロジー』（20 ページ）で説明したトポロジーと比較すると、このトポロジーには以下のメリットもあります。
  - 複数の同時ローカル・ユーザー向けの端末サービスの必要なしに、同時クライアントがサポートされます。
  - 1 台のクライアントでハードウェア障害が発生しても、サーバーやその他のクライアントに影響はありません。

– 主にクライアント層の中で稼働するコンポーネント (例えば、Blueprint Director、InfoSphere Data Architect) は、サーバー・ハードウェア障害が発生した場合でも引き続き使用可能です。

考慮事項を下記に示します。

- 小規模ではないソリューションでは、サーバー層は、特にエンジン層のマシン・リソースを獲得するために他の層と競合します。
- サーバーの障害が発生すると、エンジン層、サービス層、および Information Server リポジトリー層が完全に停止します。

2.5 層ごとに専用のコンピューターのトポロジー

図 2-4 に、このトポロジーを示します。各層に専用のコンピューティング・リソースがあります。図にはクライアント層のコンピューターが 1 台だけ示されていますが、このトポロジーに複数のクライアント層のコンピューターを組み込むことができます。

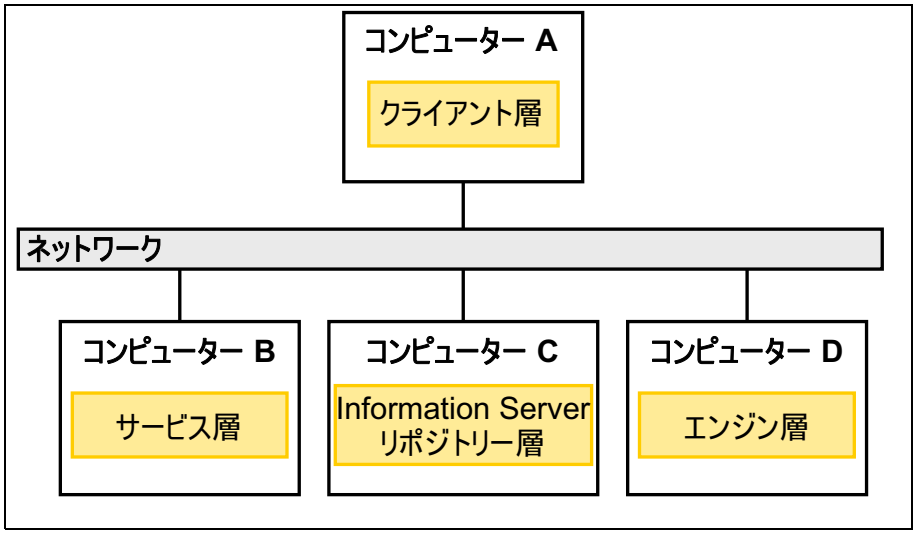


図 2-4 層ごとに専用のコンピューター

このトポロジーのメリットと考慮事項

このトポロジーには、2.4『クライアント / サーバー・インストール・トポロジー』（21 ページ）のトポロジーのすべてのメリットに加えて、以下のメリットがあります。

- 各サーバー層コンポーネントのサイジングとスケーリングは、負荷または発展する要求に応じて個別に行うことができます。
- 各サーバー層コンポーネントを個別に管理できます。これは、一部の組織ではリポジトリー (データベース) 層またはサービス (WebSphere Application Server) 層の格段の要件となる場合があります。
- 各層が負荷に応じて同じマシン・リソースを獲得するために他の層と競合することはありません。
- 前述のトポロジーと比較すると、エンジンのホスト・マシンでの障害が原因でサービス層で障害が発生することはありません。そのため、エンジン層のサーバーが失われても、一部の Business Glossary、Metadata Workbench、および FastTrack の機能は引き続き作動します。

考慮事項を下記に示します。

- サービス層または Information Server リポジトリー層のホストで障害が起こると、サーバー・サイドの全機能へのクライアントのアクセスが完全に失われ

ます。このトポロジーを拡張してこの制約を緩和する方法については、2.7『Information Server クラスター・インストール』（24 ページ）を参照してください。

- Information Server リポジトリー層とサービス層が別々のホスト上にあるこのようなトポロジーでは、これらの層の間で高帯域幅の通信リンクが維持される必要があり、これらの 2 つの特定の層が同じサブネット上になければなりません。
- この構成でのフェイルオーバー・シナリオは、若干複雑です。各層のフェイルオーバー方法を個別にセットアップする必要があるためです。

2.6 専用エンジン・トポロジー

このトポロジーでは、サービス層と Information Server リポジトリー層は 1 台のコンピューターにインストールされます。エンジン層は、別のコンピューターにインストールされます。クライアント層は、個々のコンピューターにインストールされます。

図 2-5 (23 ページ) にこのトポロジーを示します。この図には、明確にするためにクライアント層のコンピューターが 1 台だけ示されていますが、このトポロジーには通常は複数のクライアント層のコンピューターが組み込まれます。

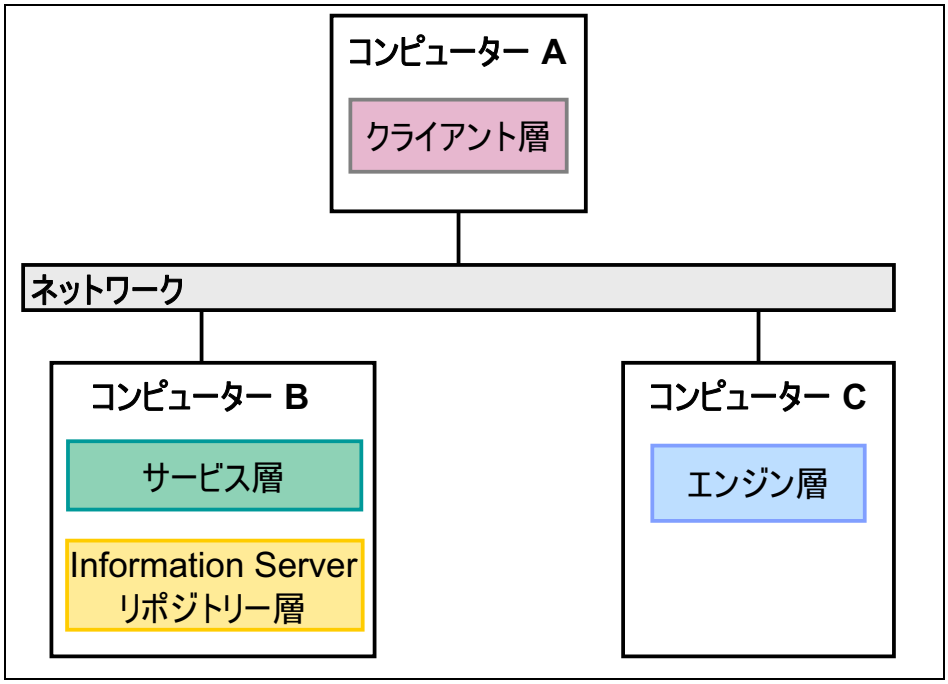


図 2-5　専用エンジン

このトポロジーのメリットと考慮事項

メリットを下記に示します。

- このトポロジーには、前述の 2.4『クライアント / サーバー・インストール・トポロジー』（21 ページ）で説明したトポロジーのすべてのメリットがあります。
- Information Server リポジトリー層をサービス層と一緒にインストールすると、層の間でネットワーク遅延が生じないため、パフォーマンスが最適になります。また、エンジン層のアクティビティーが多くなっても、サービス層と Information Server リポジトリー層のステージに影響はありません。

- IBM InfoSphere DataStage および QualityStage のジョブ要件が高く、幅広いプロファイル作成またはデータ・ルール処理で IBM InfoSphere Information Analyzer が使用される環境では、エンジン層の分離が適しています。
- エンジンのホスト・マシンで障害が発生しても、Business Glossary、Metadata Workbench、および FastTrack の大半の機能は引き続き作動します。

このトポロジーの 1 つの考慮事項は、サービスまたはリポジトリーのホストで障害が発生するとサービスが完全に失われることです。このトポロジーを拡張してこの制約を緩和する方法については、2.7『Information Server クラスター・インストール』（24 ページ）を参照してください。

2.7 Information Server クラスター・インストール

ここまでのセクション (2.3『単一コンピューター・トポロジー』（20 ページ）から 2.6『専用エンジン・トポロジー』（23 ページ）) では、最大でも層ごとに 1 台の物理サーバーを使用する方法を説明しました。このセクションでは、Information Server インストール環境のクラスタリングについて説明します。

Information Server インスタンスのクラスタリングは、層単位で、またインストール環境全体としても行います。

選択する方法に応じて、クラスタリングは、可用性の強化やスケーラビリティーの向上というメリットをもたらします。

また、Information Server のクラスタリングにより、デプロイメント・トポロジーの範囲が複数のエンジンのインストール環境、超並列処理 (MPP)、およびグリッド構成に拡大されます。

クラスター・インストールでは、次の 2 つの重要な用語が使用されます。

- *高可用性*: この用語は、ハードウェア障害発生時に素早くリカバリーできるシステムを表します。
- *災害復旧*: この用語は、災害によって 1 次サイト全体が破壊されたり、作動不能になった場合に、データ・センターを別のサイトの別のハードウェアでリカバリーできる能力を表します。

同じデータ・センター内での Information Server インスタンスのクラスタリングは、正確に構成されると、そのデータ・センター内で高可用性を提供します。Information Server の災害復旧に関する考慮事項は、2.8『災害復旧』（48 ページ）で説明しています。

2.7.1 アクティブ / パッシブ・クラスター

このセクションでは、2.4『クライアント / サーバー・インストール・トポロジー』（21 ページ）で説明したトポロジーを拡張するアクティブ / パッシブ・クラスター・アーキテクチャーについて説明します。*アクティブ / パッシブ*クラスター・トポロジーでは、2 台のサーバー・サイド・コンピューターが使用され、Information Server のバイナリーおよびデータを格納するディスク・ストレージは、これらのコンピューターの外部で管理されているストレージ上に配置される必要があります。この外部ストレージは、いずれかのサーバー・コンピューターからマウントまたはアンマウントできる必要があります。このストレージ域は、通常はストレージ・エリア・ネットワーク (SAN) デバイスを介して、または稀に Network Attached Storage (NAS) を介して管理されます。

このセクションで説明するトポロジーでは、Information Server リポジトリー層、エンジン層、およびサービス層はすべて外部ストレージ上にインストールされます。外部ストレージは、アクティブ・サーバー上にマウントされます。したがって、どの時点でも、いずれか 1 台のコンピューター (アクティブ・サー

バー) が、開始された InfoSphere Information Server インスタンスをホストします。もう 1 台のコンピューター (パッシブ・サーバー) では、オペレーティング・システムは始動されますが、Information Server インストール環境が格納されている外部ストレージはマウントされず、Information Server は始動されません。

高可用性 (HA) を提供するクラスター・ソフトウェアは各サーバーにインストールされる必要があります。

以下に必要なクラスター・ソフトウェアの例を挙げます。

- IBM Tivoli® System Automation for Multiplatforms
- IBM PowerHA®
  (旧称 HACMP™ - High Availability Cluster Multiprocessing)
- Oracle Sun Cluster
- Hewlett-Packard Serviceguard
- Microsoft Cluster Services on Windows Servers

HA クラスター・ソフトウェアは、*ハートビート* を維持します。これは、アクティブ・サーバーからパッシブ・サーバーに送信され、アクティブ・サーバーが作動可能であることを示す周期信号です。

アクティブ・サーバーでハードウェア障害が発生した場合、HA ソフトウェアは、それまでのアクティブ・サーバーから外部ストレージをアンマウントして、それをパッシブ・サーバーにマウントします。次に、パッシブ・サーバーで Information Server が再始動されます。また、HA ソフトウェアは、新しい着信クライアント接続を新しいアクティブ・ホストにトランスペアレントに経路指定します。サービスはこのように新しいアクティブ・サーバーで続行されるため、アドミニストレーターは障害を起こしたホストを処理できます。障害が修正された後、障害を起こしたホストはクラスター内のパッシブ・ホストになり、クラスターはそこから続行されます。

外部ストレージ・システムも、単一のディスク障害がトランスペアレントに処理されることを内部的に保証する必要があります。これは、SAN および NAS ストレージ・システムの標準機能です。単一ディスク障害に対処する SAN または NAS システムの機能は、一般に RAID (Redundant Array of Independent Disks) テクノロジーを使用して実現できます。

図 2-6 にこのトポロジーを示します。これまでと同様、明確にするために、図にはクライアント層のコンピューターが 1 台だけ示されていますが、複数のクライアント層のコンピューターを使用できます。

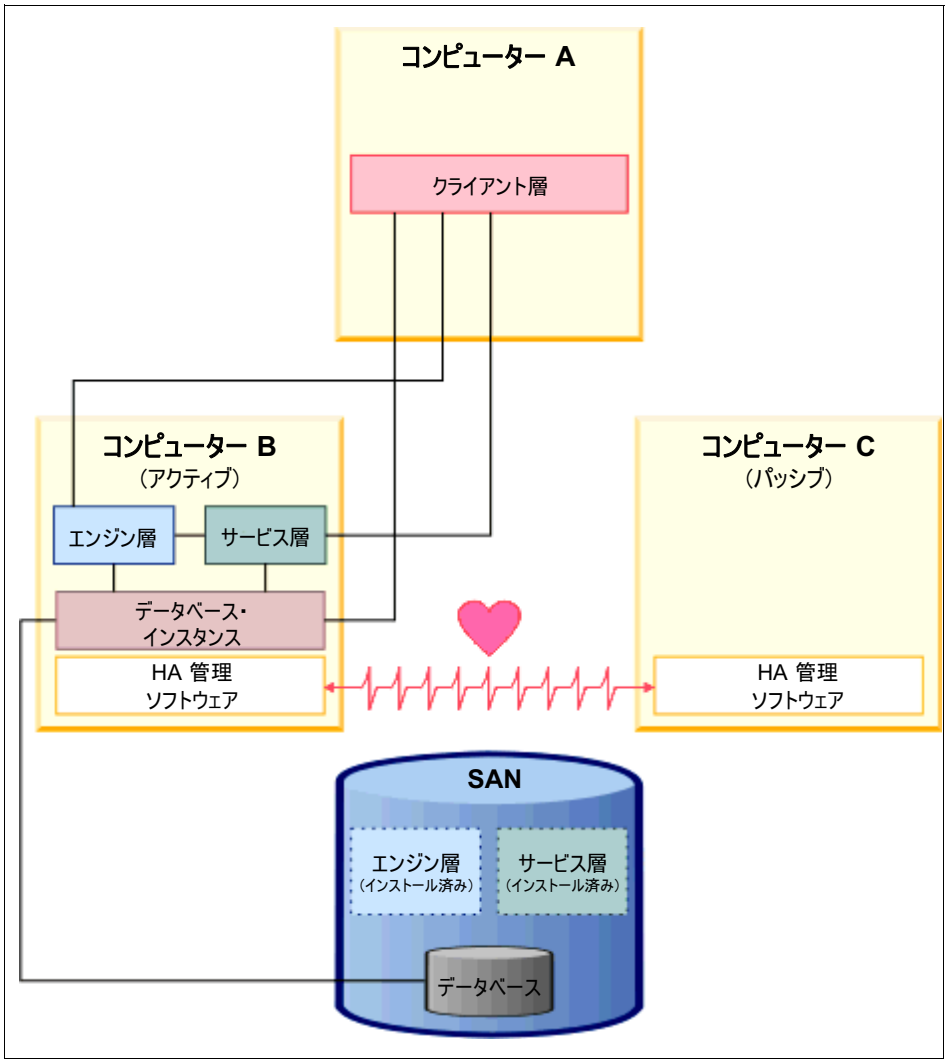


図 2-6　アクティブ/パッシブ・トポロジー

図 2-7 は、フェイルオーバー・モードでのこのトポロジーを示しています。

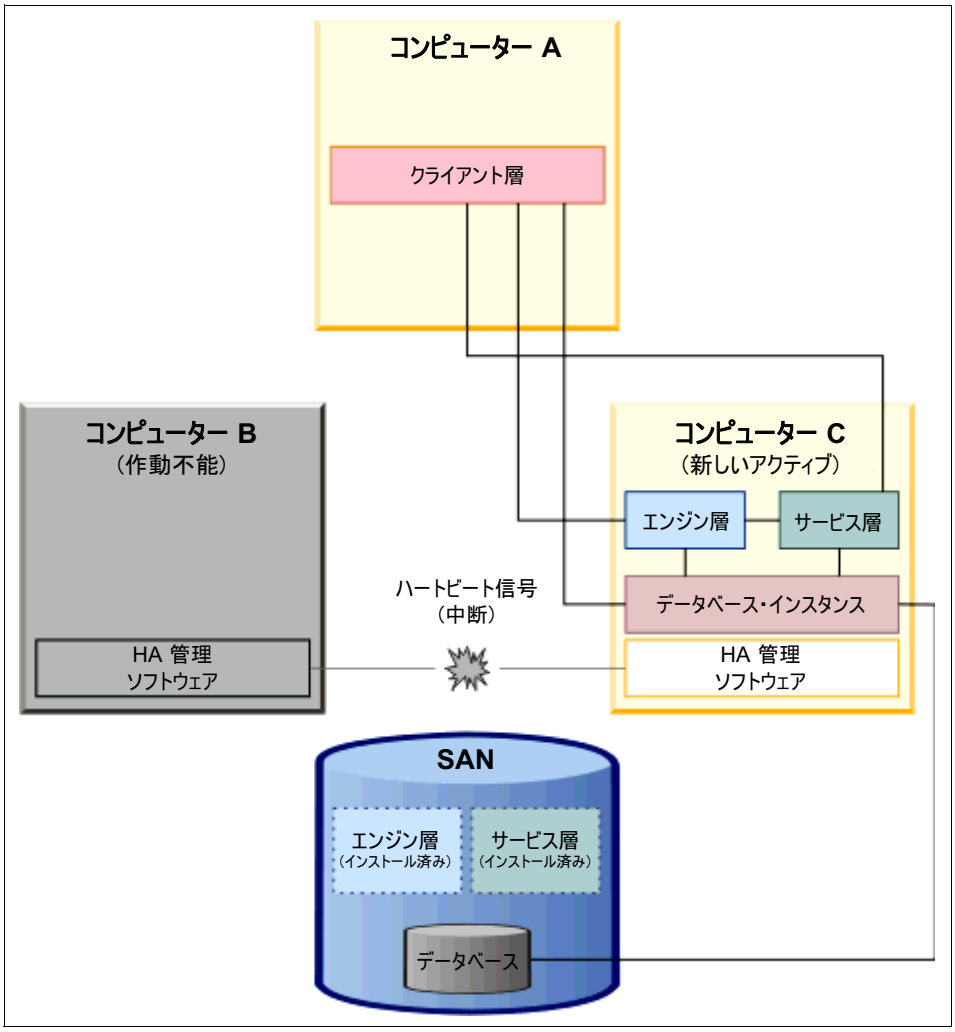


図 2-7　フェイルオーバー・モードのアクティブ / パッシブ・クラスター

クライアントが自動的に現在のアクティブ・ノードに経路指定できるように、HA 管理ソフトウェアは仮想ホスト名機能を提供する必要があります。クライアントは、クラスターへの接続ポイントとして仮想ホスト名を使用します。HA 管理ソフトウェアは、どちらのアクティブ・サーバー・コンピューターが使用中であるかに関係なく、この仮想ホスト名を維持して、内部の仮想から物理へのマッピング・テーブルを使用して着信接続を現在のアクティブ・ホストに経路指定します。

このトポロジーのメリットと考慮事項

メリットを下記に示します。

- クライアント / サーバー・トポロジーと比較すると、このソリューションは、はるかに高い可用性を提供します。サーバー層のホストのハードウェアで障害が起こっても、発生したハードウェア障害が修正されるまで停止が続くことはないためです。
- HA クラスター・ソフトウェアは、部門またはデータ・センターの基準に従って選択可能です。クラスター・ソフトウェアは、少なくとも、カスタマイズ可能なハートビート、仮想ホスト名機能、およびフェイルオーバー・シーケンスに対するアクションをスクリプト記述する機能を提供する必要があります。

- サーバー・ファームウェア・アップグレードのオペレーティング・システム・パッチを適用する必要があるシステム・アドミニストレーター向けに、停止を低減するための追加のオプションが使用可能です。

考慮事項を下記に示します。

- インストールが若干複雑になります。
- スタンバイ・ホストは常に完全にアイドル状態になります。この制約は、2.7.6『アクティブ / アクティブ・エンジン層のトポロジー』（40 ページ）で説明しているアクティブ / アクティブ・トポロジーを使用することで緩和できます。
- フェイルオーバー・スクリプトでは、フェイルオーバー・ノード上のアプリケーション・サーバーの始動に数分間かかる可能性があるため、フェイルオーバー中の停止期間はこの時間と一致します。
- このトポロジーは、開発と本番のインストール環境でデプロイできます。ただし、開発環境でフェイルオーバーが行われ、フェイルオーバー時にクライアントで設計中の変更が開かれていて保存されていない場合、フェイルオーバー・ノード上の Information Server の再始動操作の後でこれらの変更を再入力する必要があります。
- 基本的なクライアント / サーバー・トポロジーよりもスケーラビリティーが向上することはありません。

本章の後続のセクションでは、上記の制約をさらに改善できる層単位のレベルでのクラスタリング・オプションについて詳しく説明します。各層を別々に検討する場合、配置が複雑化する可能性がありますが、結果としてメリットも得られます。メリットと考慮事項については、後続のセクションで説明します。

2.7.2 サービス層のクラスタリング

図 2-8 (29 ページ) に、WebSphere Application Server のサービス層の中でホストされる幅広い Information Server の機能の概要を示します。WebSphere Application Server を稼働するホストが 1 台である場合、この層で障害が発生すると、これらのバックボーン・サービスが失われるために Information Server が完全に停止する可能性があります。障害の原因として、ホスト・マシンのハードウェア障害が考えられます。同様に、これらのサービスの処理負荷によって使用可能なマシン・リソースが使い尽くされた場合は、サービス全体が低下するか、ユーザーへの応答時間が長くなる可能性があります。

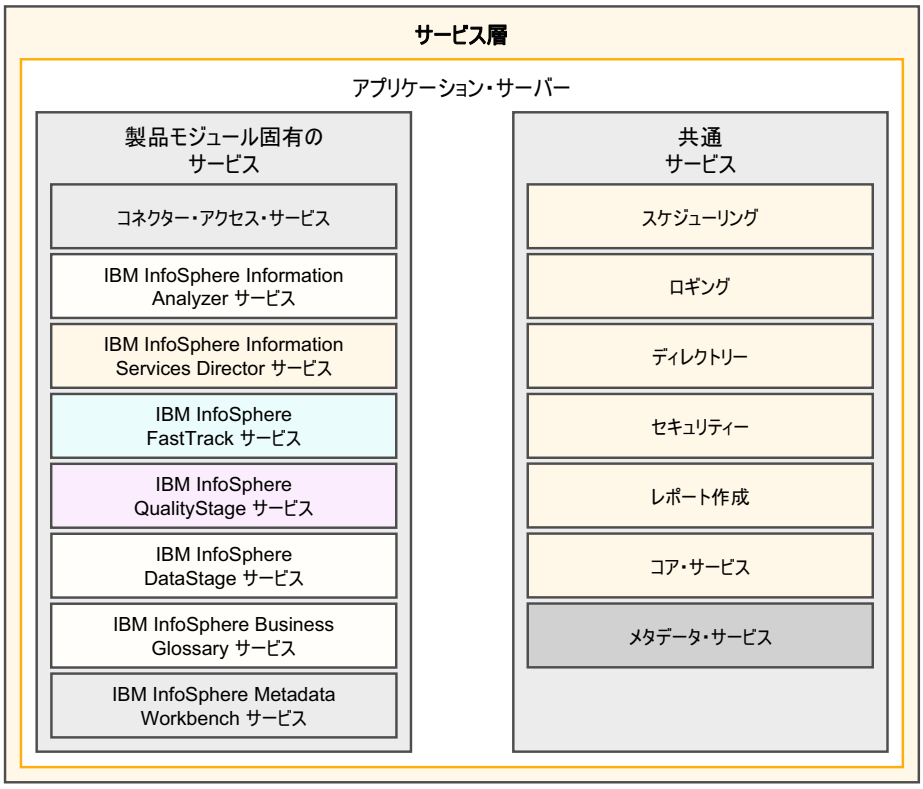


図 2-8　WebSphere Application Server でホストされる機能

Information Server バージョン 8.5 以降では、製品インストールに、WebSphere Application Server Network Deployment エディションがバンドルされています。このバージョンのアプリケーション・サーバーは、サービス層のロード・バランシングと高可用性アーキテクチャーの両方をサポートするために使用できる機能を備えています。

WebSphere Application Server Network Deployment は、アクティブ / アクティブ・クラスターと、インストール・ワークロード全体が管理される N 個の冗長ノードのアレイもサポートします。さらに多くのノードを追加してサービス層の完全なスケーラビリティーを得ることができるため、WebSphere Application Server 内でホストされる Information Server の機能の側面も同時ユーザーの要件とリソース要求に応じて拡張できます。

この機能を利用するために、WebSphere Application Server Workload Manager (WLM) プラグインが搭載されたフロントエンド Web サーバーまたはロード・バランサー、あるいは両方がデプロイされます。これらのフロントエンド・コンポーネントは *フロントエンド・ディスパッチャー* と呼ばれます。

Web サーバー・フロントエンドの使用

図 2-9 は、WLM プラグインが搭載されたフロントエンド Web サーバーを示しています。サポートされる Web サーバーは、IBM HTTP Web Server または The Apache Software Foundation 提供の Apache Web Server (Apache) のいずれかです。単一障害点を回避するため、1 次 Web サーバーで障害が起こった場合に完全かつ迅速にテークオーバーを実行できるようにバックアップ Web サーバーを使用してください。

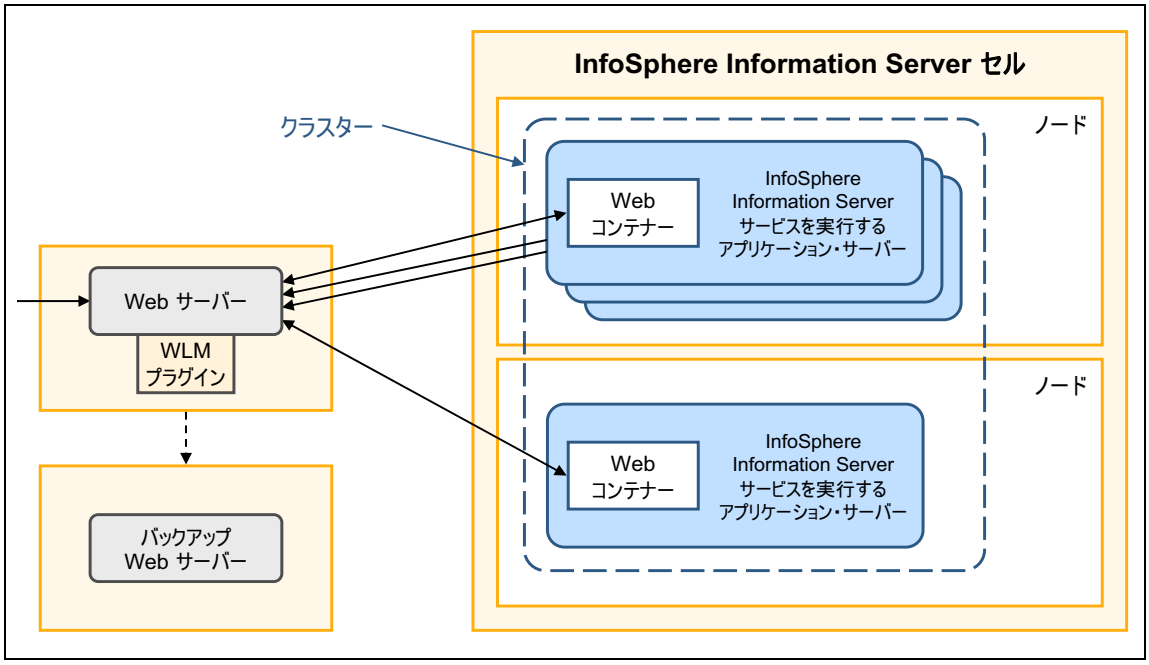


図 2-9　Web サービスを使用するサービス層のロード・バランシング

開発環境では、Web サーバーを、WebSphere Application Server と同じサーバーに配置できます。本番環境では、セキュリティーに関する考慮事項から、Web サーバーと WebSphere Application Server の間にファイアウォールが必要になる場合があります。その場合、別々のサーバーにインストールできます。

このタイプのクラスタリングされた WebSphere Application Server インストール環境では、クライアントが WebSphere Application Server 内で接続して新規セッションを作成すると、定義済みのワークロード管理アルゴリズムに従ってセッションがノードに割り振られます。特定のサーバー内でセッションが作成された後、セッションはそのノード内で完了時まで実行されます。WebSphere Application Server はセッションをすべての WebSphere Application Server クラスター・メンバーに複製するため、クラスター・メンバーで障害が起こった場合に、最低限の停止時間 (通常は数秒間) でセッションを別のクラスター・メンバーで再開できます。

ロード・バランサー・フロントエンドの使用

Web サーバーを使用する代わりに、ロード・バランサーを使用できます。図 2-10 に、このアプローチを使用するサービス層のトポロジーを示します。

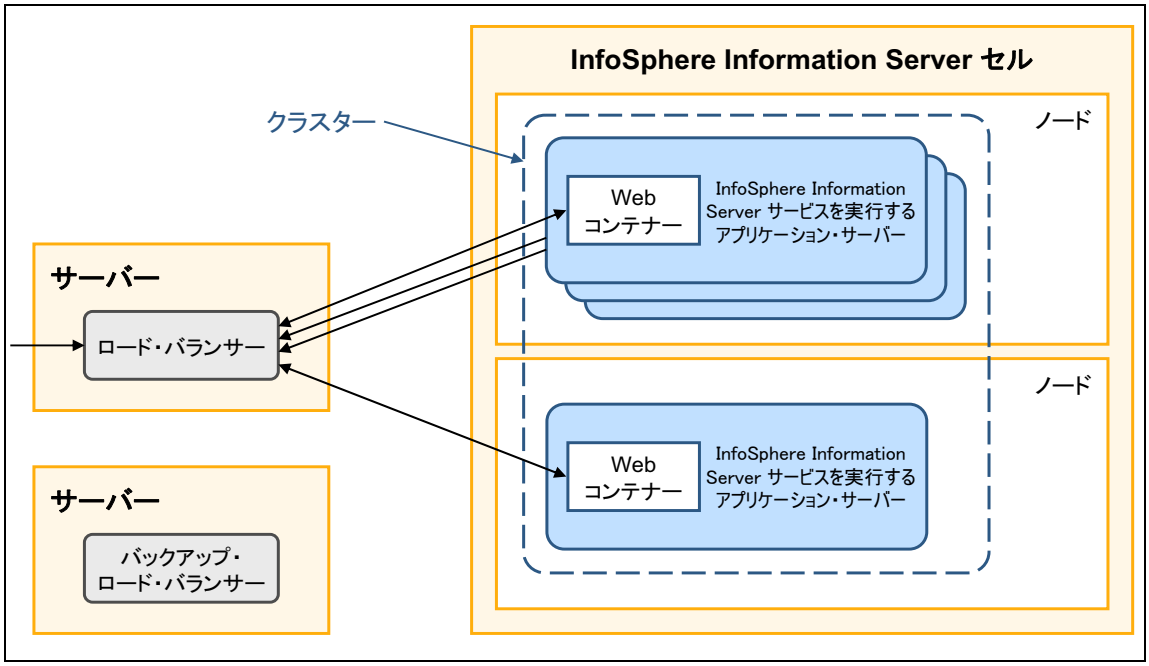


図 2-10　明示的なロード・バランサーを使用するサービス層

ロード・バランサーは、ハードウェアまたはソフトウェアのロード・バランサーを使用することによって実装できます。唯一の要件は、選択するロード・バランサーが WebSphere Application Server のサポートを宣言する必要があることです。

- ハードウェア・ロード・バランサーの一例として、BIG-IP (F5 Networks, Inc 提供) という名前の製品ファミリーが挙げられます。
- ソフトウェア・ロード・バランサーの一例としては、IBM WebSphere Edge Components for WebSphere Application Server が挙げられます。

ロード・バランサーの選択は、組織のデータ・センターまたはアーキテクチャーの基準によって決定される場合があります。ただし、前に説明した Web サーバーの WLM プラグインと比較すると、ロード・バランサーを使用する場合はワークロード管理が複雑になる可能性があります。

ロード・バランサーのセットアップは比較的複雑になる場合があります。例えば、ロード・バランサーではセッションのアフィニティーのセットアップが複雑で、場合によってはネットワーク専門家の関与が必要になります。ハードウェア・ロード・バランサーには高額なコストがかかる可能性もあります。

望ましいアプローチは、ロード・バランサーのセッション・アフィニティー・タイムアウトが、WebSphere Application Server で Information Server セッションとして設定されているのと同じ値に維持されることです。デフォルトでは、この値は 30 分です。DataStage 開発環境では、DataStage リソースを解放するためのタイムアウトとして DataStage Administrator で設定されている値と一致させるために、この値が増える可能性があります。

アフィニティー・タイムアウト値で指定されている非アクティブ期間が経過した後、ロード・バランサーは、ロード・バランシング・アルゴリズムを再び使用することにより、クライアントからの後続の要求の割り当てまたは再割り当てを行います。

ロード・バランサーおよび Web サーバー・フロントエンドの使用

フロントエンドは、ロード・バランサーと Web サーバーの組み合わせで構成される場合もあります。高可用性と、サーバーおよびワークロード容量の最大限のバランスを得るには、ロード・バランサーを Web サーバー群のアップストリームにデプロイします。この方法は *IP スプレイヤー・アーキテクチャー*と呼ばれます。

ロード・バランサーは、サーバーの可用性とワークロード容量に基づいて Web サーバー間でインテリジェントなバランシングを実行します。Web サーバー・レベルで単一障害点をなくすには、このトポロジーを選択します。また、この配置では、複数の Web サーバー間で Web 要求の負荷が分散されます。Information Server は、IBM およびその他のベンダーが製造する多くの IP スプレイヤー・ソリューションをサポートします。

ロード・バランサー・レベルで単一障害点を防止するには、アクティブなロード・バランサーで障害が起こった場合にテークオーバーするバックアップ・ロード・バランサーをデプロイします。

図 2-11 (32 ページ) に、IP スプレイヤー・トポロジーを示します。この図には、2 台のフロントエンド Web サーバーを使用する InfoSphere Information Server クラスターが示されています。ロード・バランサーは、Web サーバーのアップストリームにデプロイされています。バックアップ・ロード・バランサーもデプロイされています。Web サーバーとロード・バランサーは別々のサーバー上にインストールされています。

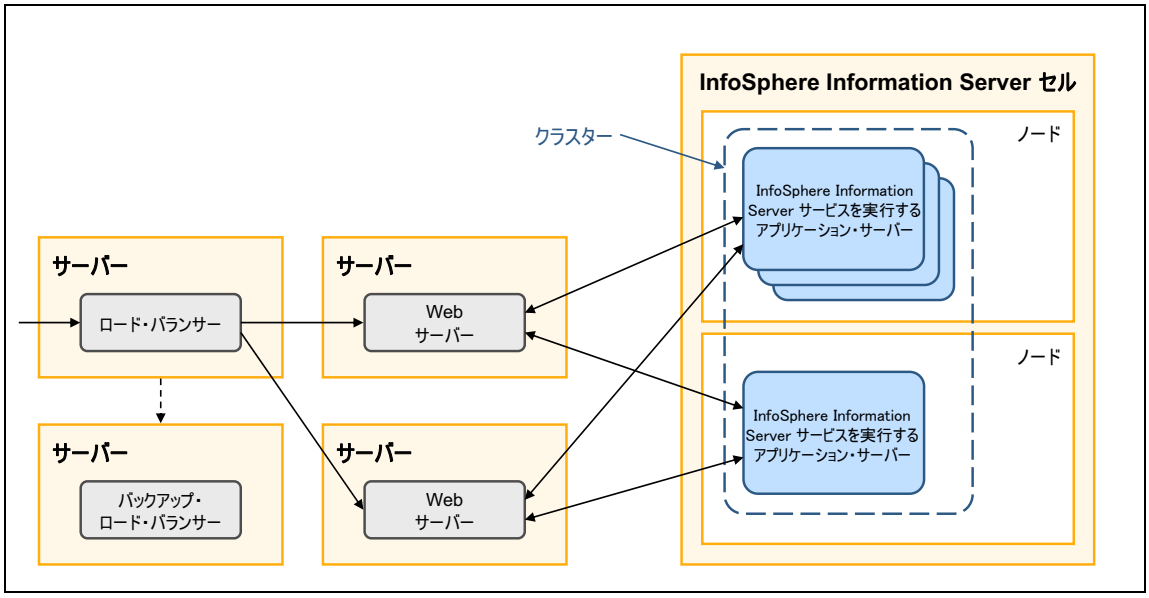


図 2-11　IP スプレイヤーのフロントエンド

サービス層のクラスタリングのメリットと考慮事項

すべてのサーバー層を一緒にクラスタリングする方法については、2.7.1『アクティブ / パッシブ・クラスター』（24 ページ）で説明しています。このオプションは可用性をもたらしますが、フェイルオーバー・プロセス時に完全に再始動するまでの停止期間は 10 分間から 15 分間になります。また、このオプションでは、その時間の 100% にわたり、サーバーがアイドル状態になる必要があります。

サービス層だけを考慮すると、スイート全体のアプローチと比較して、サービス層のクラスタリングには、この層でホストされている Information Server の機能のスケーラビリティー、ロード・バランシング、および可用性の点で大きなメリットがあります。サービス層のフェイルオーバーにかかる時間は数秒間で、

サービス層インフラストラクチャーに関与するすべてのサーバーは通常動作時に Information Server インストール環境全体の代わりに処理を実行できます。

これらのメリットを得ようとする場合、関連するサーバーの初期セットアップと構成がさらに複雑になります。また、Web サーバー、ロード・バランサー (ハードウェアまたはソフトウェア) あるいは両方を使用してフロントエンド・ディスパッチャーの機能を追加するという要件があります。

ライセンス交付の計算 : ライセンス交付の計算では、すべてのアクティブなプロセッサー・バリュー・ユニット (PVU) が考慮されます。(比較すると、アクティブ / パッシブ・クラスターではアクティブ・サーバーのみの PVU が計算されます。)

2.7.3 Information Server リポジトリー層のクラスタリング

Information Server リポジトリー層のデータベース・クラスターを作成できます。

Information Server は、IBM DB2、Oracle Corporation Oracle Database、または Microsoft SQL Server を使用して実装される Information Server リポジトリー層をサポートします。Information Server のインストーラーには、Information Server リポジトリー・データベースとして使用するために無料で付与される IBM DB2 のライセンスが含まれています。使用する Information Server リポジトリー層データベース・テクノロジーの選択は、お客様が選ばれるデータベース・テクノロジーとローカル・データベース管理スキルの可用性によっても決定されます。

クラスタリングの実装の正確な詳細は、Information Server リポジトリー層で採用されているデータベース・テクノロジーによって異なります。ただし、高水準での詳細はすべて似ています。すなわち、冗長性のためにデータベースをホストする 2 台以上の物理サーバーが必要です。データベースのクラスター機能がサーバーの停止に対処します。その際、Information Server の接続は残りのデータベース・サーバーに送信されるという実質的な影響があります。

このセクションの残りの部分では、選択するデータベース・テクノロジー (DB2、Oracle Database、または SQL Server) に応じて Information Server リポジトリー層のクラスタリングに使用できる詳細情報とオプションについて説明します。

DB2 Information Server リポジトリー層のクラスタリング

Information Server は、DB2 Information Server リポジトリー層のクラスタリングで以下の選択肢をサポートします。

- アクティブ / パッシブ・クラスター
- HADR クラスター

アクティブ/ パッシブ・クラスター

この構成では、Information Server リポジトリー・データベースはクラスター内のノード間で共有されます。フェイルオーバーが行われると、クラスター内のもう 1 つのノードが DB2 の機能を提供します。高可用性を得るために、1 台のコンピューター上に単一のアクティブな DB2 インスタンスを配置して、他のコンピューター上に 1 つ以上のパッシブ・インスタンスを配置したアクティブ / パッシブ構成でクラスターをセットアップします。DB2 インスタンスで問題または障害が起こった場合、パッシブ・インスタンスがテークオーバーできます。

この構成を管理するために、DB2 には Tivoli System Automation クラスター・ソフトウェアのすべてのコンポーネントが標準搭載されています。この機能は、バージョン 9.5 以降の DB2 に組み込まれており、Information Server ライセンスに含まれています。

Tivoli System Automation ソフトウェアは、クラスター内のノード間のハートビート信号を維持します。アクティブ・ノードでハートビートが失敗すると、ソフトウェアは、パッシブ・ノードへのフェイルオーバーを開始します。この構成は、2.7.1『アクティブ / パッシブ・クラスター』（24 ページ）で説明している構成と似ています。主な違いは、クラスターが DB2 でホストされている Information Server リポジトリー層にのみ関連していることです。

この構成では、DB2 のフェイルオーバーは自動的に行われますが、新しいインスタンスがリソースを獲得して、特定のトランザクションを繰り返し、その他のトランザクションを取り消すまでに数分間かかります。中断と手操作による介入を最小限に抑えるには、DB2 自動クライアント転送 (ACR) を構成します。この機能により、Information Server インスタンスのその他のコンポーネント (WebSphere Application Server JBDC 接続など) は自動的に新しい DB2 インスタンスに再接続します。

この構成は、データベース自体の災害復旧は提供しません。その代わり、データベース・クライアント・プロセスの高可用性を提供して、新しいノードへの再接続を円滑に行えるようにします。データベース自体の冗長性を得るには、DB2 の高可用性災害復旧 (HADR) 機能を実装します。

HADR クラスター

DB2 の高可用性災害復旧 (HADR) 機能は、部分的および完全なサイト障害に対するものを含む Information Server リポジトリー層の高可用性および災害復旧設計の一環として使用できるデータベース・ログ複製機能です。HADR は、ソース (1 次) データベースからターゲット (スタンバイ) データベースにデータ変更を継続的に複製することにより、データ損失から保護します。元の 1 次データベースを使用していたクライアントは、アプリケーションの自動クライアント転送 (ACR) と再接続ロジックを使用することにより、スタンバイ・データベース (新しい 1 次データベース) に送信されます。

DB2 HADR は、ログ配信ソリューションに似た方法で機能しますが、すべての処理は DB2 内部でソフトウェア・レベルで行われます。DB2 HADR 1 次データベースは、データベース・ログ・バッファーを HADR スタンバイ・データベースに配信するために内部プロセスを使用します。次に、スタンバイ・サーバーでのプロセスにより、ログ・レコードがスタンバイ・データベースで直接的に再生されます。1 次データベースの障害発生時にフェイルオーバーが行われる場合、スタンバイ・データベースを 1 次データベースに変換して Information Server からアクセスできるようにすることができます。

HADR は、Tivoli System Automation のクラスター・ソフトウェア機能と並行して使用します。1 次データベースの障害発生時に、Tivoli System Automation クラスタリング・ソフトウェアにより、自動化された HADR テークオーバーが追加されます。Tivoli は、1 次データベース・サーバーが停止していないかどうかをモニターする機能を追加して、1 次データベース・サーバーの環境が停止していないかどうかをモニターします。図 2-12 (35 ページ) に示すように、1 次データベースがアクティブでなくなっても、スタンバイ・データベースはログ・レコードが転送されるのを待機します。クラスター・ソフトウェアを使用しない場合は、HADR のペアをモニターして、1 次データベース・サーバーの障害発生時に適切なテークオーバー・コマンドを手動で実行する必要があります。1 次データベース・サーバーの障害発生時には、HADR テークオーバーを自動化するためのクラスタリング・ソフトウェアが必要です。

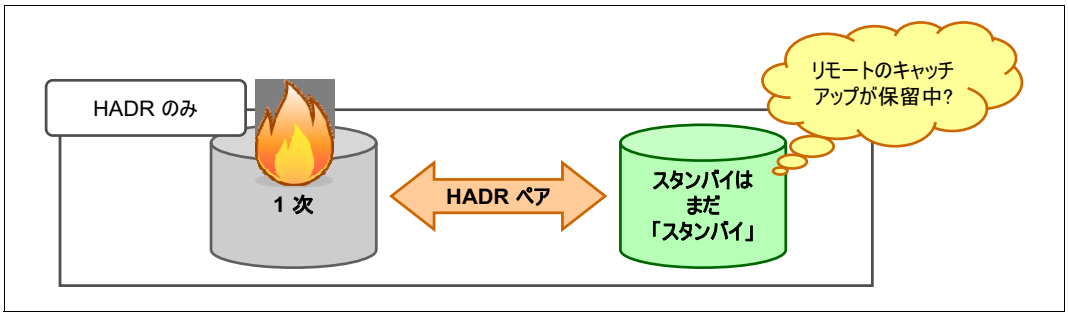


図 2-12　待機を続ける HADR スタンバイ

バンドルされている Tivoli System Automation クラスター・ソフトウェアは、テークオーバー・コマンドをスタンバイ HADR ノードに対して実行するように機能的にセットアップできます。図 2-13 に示すように、このノードは自動的に 1 次になることができます。

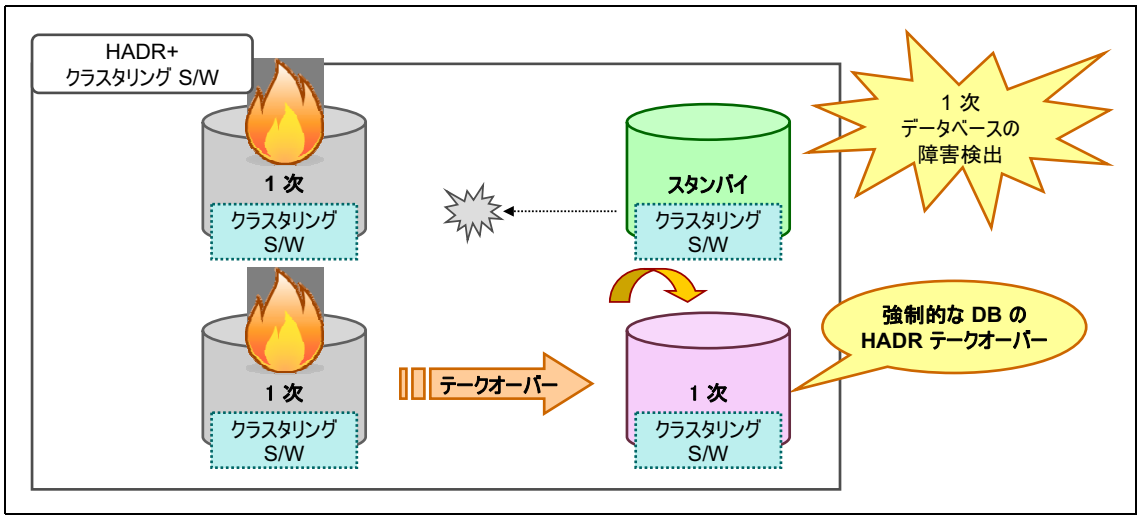


図 2-13　HADR テークオーバー・コマンドを実行するクラスター・ソフトウェア

このトポロジーのメリットと考慮事項

メリットを下記に示します。

- 高速フェイルオーバー (通常は 1 分未満) のための機能が存在します。
- パフォーマンスに対する影響はわずかです。
- サーバーのリサイクルを必要とする非メジャー・バージョン間または変更間では、サービスを中断しないローリング・アップグレードが行われます。その他のアップグレードの場合、変更に必要な時間は短縮されます。
- スタンバイ・データベースを読み取り専用モードで使用できます。読み取り専用モードは、管理またはサポートに役立ちます。
- スタンバイ・データベースをリモート・データ・センターに配置できます (2.8『災害復旧』（48 ページ）を参照)。

考慮事項は、アクティブ / パッシブ・クラスターと比較するとセットアップが若干複雑になることです。

Oracle Information Server リポジトリー層のクラスタリング

Oracle Corporation は、Oracle データベースのクラスタリングおよび高可用性のためのソフトウェアを提供しています。使用できるオプションは、Oracle Real Application Clusters および Oracle Data Guard です。

Oracle Real Application Clusters (RAC)

Oracle RAC は、複数のコンピューターが同時に Oracle Relational Database Management System を稼働できるようにするとともに、単一データベースへのアクセスを提供します。外部クライアントは、RAC に関与するどのサーバーにも接続でき、調整された一連のデータにアクセスできます。

データベース・クラスターに関与するすべてのコンピューターが同じデータをクライアントに提示する必要があるため、データベース・データの変更がクラスター全体で表示されることをシステム全体で保証して、すべてのクライアントがデータの現行バージョンを表示できるようにする必要があります。Oracle は、この機能を提供する RAC 機能を *Cache Fusion* と呼んでいます。この機能を実現するために、Oracle RAC クラスターでは、関与する RAC サーバー間に専用の高速内部ネットワークが必要です。

Oracle は、クラスター内のサーバーで障害が起こった場合は Oracle データベースがクラスター内の残りのサーバーで稼働を続行すると助言しています。また、データベース・クライアントが稼働を続行している間に保守のために個々のサーバーをシャットダウンできます。

Oracle Database バージョン 11g リリース 2 (本書の執筆時点で Information Server がサポートしている 11gR2) 以降では、ワークロード管理も組み込まれています。この場合、サーバー・プール全体で接続のロード・バランシングが行われ、Oracle Database 11g Automatic Workload Repository 機能によってサービスごとにパフォーマンスが追跡されます。

Oracle Real Application Clusters の詳細については、Oracle の資料を参照してください。

Oracle Data Guard および Oracle Active Data Guard

Oracle Corporation が Oracle Data Guard として市販しているソフトウェアは、Oracle RDBMS の拡張版を形成し、Enterprise Edition で使用可能です。「2 次スタンバイ」の役割のデータベースと「1 次」データベースの役割の代替リポジトリーの設定と維持を支援します。データベースは、1 つの役割から他の役割に移行できます。

Data Guard は、物理スタンバイと論理スタンバイの両方のサイトをサポートします。

- 物理スタンバイ・データベースは、Oracle Net ネットワーク層で 1 次データベースの正確な内容を複製します。
- 論理スタンバイ・データベースは、1 次データベースで生成された再実行をデータと SQL に変換して、それらの SQL トランザクションを論理スタンバイで再適用します。

Oracle Active Data Guard は、Oracle 11g 構成の Oracle Data Guard の物理スタンバイ機能を拡張して、1 次ノードからトランザクションをアーカイブするのと同時にスタンバイ・ノードでの読み取り専用アクセスを可能にします。

Microsoft SQL Server Information Server リポジトリー層のクラスタリング

データベース・テクノロジーとして SQL Server を選択するお客様のために、Information Server は、SQL Server でも Information Server リポジトリー層をサポートします。本書の執筆時点で、Information Server 8.7 は、SQL Server Enterprise Edition 2005 SP2 または 2008 をサポートしています。

Windows サーバーのクラスタリングのために、Microsoft は、Windows Server Failover Clustering (旧称 Microsoft Cluster Service (MSCS)) というテクノロジーを提供しています。

SQL Server で個別の Information Server リポジトリーをセットアップできます。SQL Server データベースは、Microsoft Cluster Service または Windows Server Failover Clustering を使用するアクティブ / パッシブ・クラスターでホストされます。アクティブ / パッシブ・トポロジー・クラスターの基礎の説明については、2.7.1『アクティブ / パッシブ・クラスター』（24 ページ）を参照してください。

Microsoft SQL Server クラスターの詳細については、Microsoft の資料を参照してください。

2.7.4 エンジン層のクラスタリング

Information Server の並列エンジンは、単一サーバーから数百台のサーバーまで、さまざまなクラスター・タイプをサポートする強力かつ柔軟なアーキテクチャーを備えています。

エンジン層のクラスタリングのオプションを十分に活用するには、以下の関連する並列エンジンの機能の概要を理解することが有用なアプローチとなります。

- パイプライン並列処理
- パーティション並列処理
- 並列ジョブ構成ファイル

並列エンジンは、パイプライン並列処理とパーティション並列処理の両方をサポートします。並列処理は、各ジョブ内で作動し、ジョブのデフォルトの動作となっています。

パイプライン並列処理およびパーティション並列処理

ジョブは、多数のステージで構成されます。ステージの例として、ルックアップ、結合、および集約が挙げられます。

パイプライン並列処理 は、ジョブのステージがパイプライン内のデータを処理できるエンジンの機能です。この機能により、ジョブのダウンストリーム・ステージはアップストリーム・ステージの出力を処理でき、同時にアップストリーム・ステージが次の行に移動するため、すべてのステージが同時に処理を続行します。

パーティション並列処理 は、各ジョブ・ステージの複数のインスタンスがジョブの開始時に作成されるものとして概念的に理解されるエンジンの機能を記述するために使用されます。ジョブが実行されると、データは複数のセットに分割され、各ステージ・インスタンスが割り振られたデータのセットを処理します。各ステージ・インスタンスは、他のインスタンスが割り振られたセットを処理している間に並行してデータのセットを処理します。データのセットは*パーティション*と呼ばれます。データを複数のセットに分割するプロセスは*データのパーティション化*と呼ばれ、対応するステージ・インスタンスの数は*パーティション化の度合い*と呼ばれます。

パイプライン並列処理とパーティション並列処理の両方を自動的に実行すると、データを数倍も高速に処理できます。図 2-14 に、変換、強化、およびロードの 3 つのステージで構成される概念的なジョブを示します。

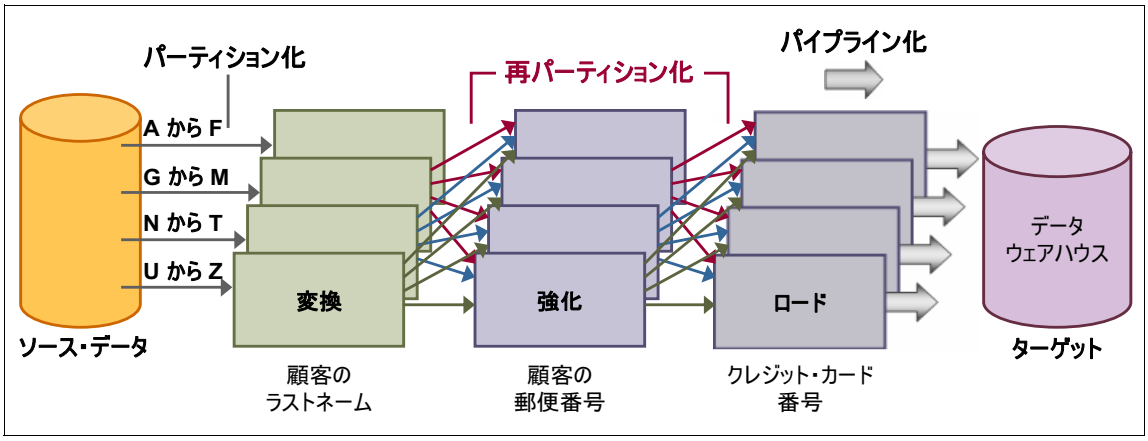


図 2-14　パーティション並列処理とパイプライン並列処理

3 つのステージはすべてパイプライン内で実行されるため、ダウンストリームの行がターゲットのウェアハウスにロードされるのと同時に、アップストリームの変換と強化のステージにより、ソースから抽出された後の行が処理されています。同じ図のジョブは、パーティション化の 4Way の度合いも示しています。各オペレーターの 4 つのインスタンスが示されています。

この例では、顧客のラストネームの先頭文字でデータのパーティション化を行う変換オペレーターが示されています。そのため、変換ステージの 1 つのインスタンスは、ラストネームが A から F で始まる顧客を処理しています。同時に、2 番目のインスタンスは、ラストネームが G から M で始まる顧客を処理しており、同様に続きます。

パイプライン並列処理とパーティション並列処理を組み合わせることで、この概念的なジョブは、少なくとも 20 倍高速にデータを処理します。

並列エンジン構成ファイル

並列エンジン・ジョブは、実行時に割り当てられる*構成ファイル*をサポートします。ジョブごとに最大で 1 つの構成ファイルがあります。

構成ファイルは、各ジョブの実行に使用されるパーティションの数を記述します。各パーティションは、構成ファイル内で実行場所のサーバーとディスクやその他のリソースへの参照によって記述されます。各パーティションは、構成ファイルでノードとして参照されます。そのため、図 2-15 (39 ページ) では、4 つのノードが含まれた構成ファイルが使用されています。

構成ファイルは、各ノードについて、処理中のデータと一時 (スクラッチ) 領域を保持するために使用されるディスクの領域と、ノードが使用すべきコンピューター・ホストを記述します。そのため、ジョブのすべてのパーティションを 1 台のサーバーで実行するか、各ノードを別々のサーバーで実行して、ディスクの同じまたは異なる領域、あるいは任意の順列を使用できます。

この強力な機能は、ジョブに関連付けられている実行時並列構成ファイルを変更するだけで、Information Server エンジン・リソースを必要とするジョブのスケールアップまたはスケールダウンをワークロードに合わせて容易に行えることを意味します。その際、その他の変更は必要なく、ジョブの再コンパイルも必要ありません。また、この機能は、構成ファイルを変更するだけで、ジョブを別のサーバーで実行したり、クラスター内のさまざまな数のサーバーを使用できることも意味します。

図 2-15 に、ソース・システムの表からの情報を変換して、集約した後、結果として生成されるデータをデータウェアハウスの表にロードする概念的な DataStage ジョブを示します。

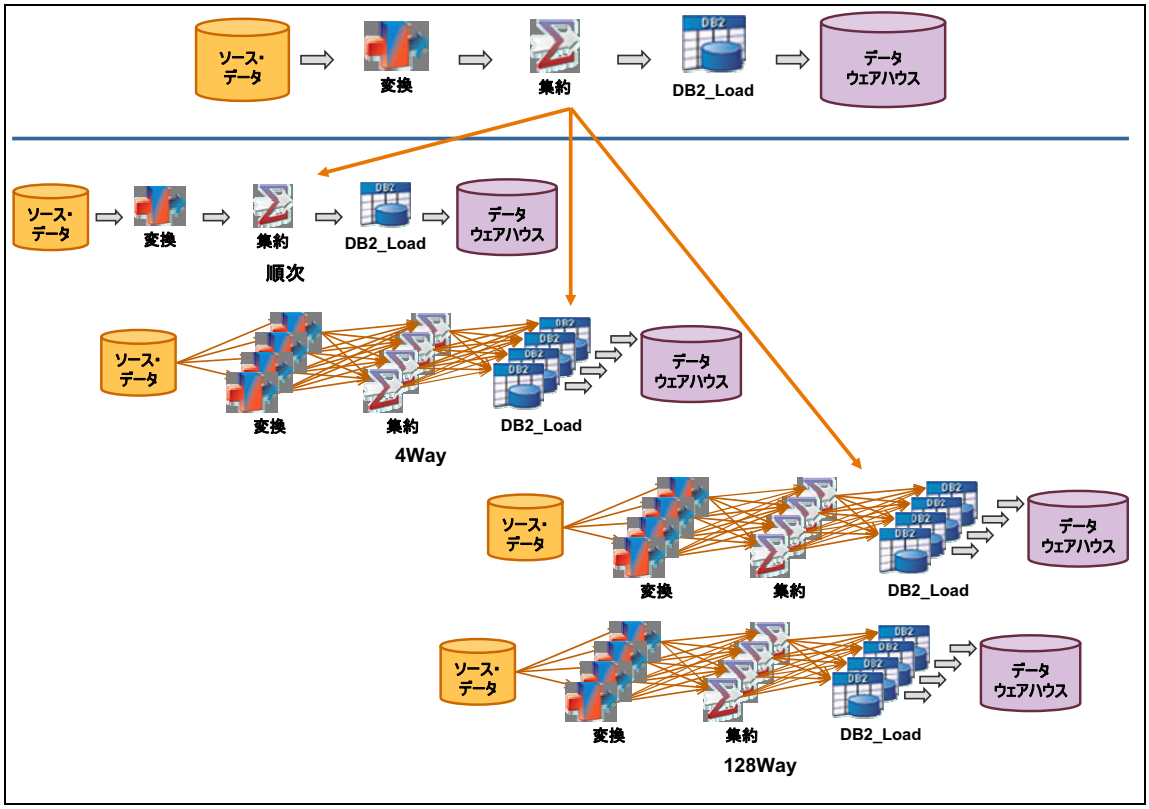


図 2-15　実行時に選択可能な並列処理

この図は、同じジョブをパーティション化のさまざまな度合いで実行できることを示しています。そのためには、関連する構成ファイル (この場合は順次 (1 ノード)、4Way のパーティション化 (4 ノード)、128Way のパーティション化 (128 ノード)) を選択します。

パイプライン並列処理は、ジョブによって使用されている構成ファイル内のノードの数に関係なく実行されます。

プロジェクトに要件に基づいて、数十から数百のこのようなジョブで統合ソリューションが構成される場合があります。また、お客様は単一プラットフォーム上に複数の統合ソリューションをデプロイすることが多いため、1 つ以上のプロジェクトでジョブの総数が数千になる場合があります。制限は、ハードウェア・プラットフォームの容量によって決まります。

ノードを柔軟にジョブに割り当てることができ、並列処理の度合いも柔軟にジョブ単位で指定できるため、構成ファイルを使用すると、他の方法では実現できないソリューションのワークロード管理、スケーラビリティー、およびクラスター設計のオプションを利用できます。

2.7.5 アクティブ / パッシブ・エンジン層のトポロジー

このトポロジーは、2.7.1『アクティブ / パッシブ・クラスター』（24 ページ）で説明したオプションと似ています。相違点は、このオプションでは、エンジン層が他の層から独立した独自のアクティブ / パッシブ・クラスターにインストールされることです。その他の層は、前のセクションで説明したように個別にクラスタリングすることができます。

前述同様に、クラスター・ソフトウェアは、現在のアクティブ・ホストへの仮想ホスト名のポインティングを提供します。このオプションでは、並列エンジン構成ファイルは現在のアクティブ・サーバーでのパーティション並列処理をサポートし、ノード名は仮想ホスト名に設定されます。

このトポロジーのメリットと考慮事項

メリットは、エンジン層の高可用性です。

考慮事項を下記に示します。

- 現在パッシブになっているノードは、どちらのサイドであっても、エンジンのワークロードに使用されません。
- サーバー障害の発生時に実行されていたジョブは、再開される必要があります。多数のジョブで構成される典型的な統合ソリューションは、最初から最後まで順々に実行される場合、1 時間程度にわたって通常のオペレーターで実行される可能性があります。ソリューションでは、障害発生時点から再開してフェイルオーバー中の停止の影響を最小限に抑えるために、DataStage シーケンサーのチェックポイント再開機能を使用する必要があります。

2.7.6 アクティブ / アクティブ・エンジン層のトポロジー

このセクションでは、2.7.1『アクティブ / パッシブ・クラスター』（24 ページ）および 2.7.5『アクティブ / パッシブ・エンジン層のトポロジー』（40 ページ）で紹介したアクティブ / パッシブ・アーキテクチャーの拡張版について説明します。

このトポロジーは、セットアップの点ではアクティブ / パッシブと似ていますが、並列エンジン構成ファイルを使用してクラスター内の両方のエンジン・ノードでエンジン・ジョブを実行できる点が主な違いです。

アクティブ / パッシブとは対照的に、このトポロジーでは、エンジン層のソフトウェアおよびアプリケーション・データ領域をエンジン層のクラスター内の両方のサーバーに同時にマウントする必要があります。そのため、クラスター・ソフトウェアとファイル共有ソフトウェア (クラスター・ファイル・システムまたは IBM GPFS ソフトウェアなど) の両方をインストールする必要があります。

複数のエンジンが関与する場合、インストールのメリットがあります。ただし、エンジンの単一インストールを実行する要件はあります。インストール・イメージを共有できるため、この単一エンジン・インストールは、クラスター内のノードを介して実行できます (仮想ホスト名を指定する必要があります)。

このトポロジーでは、エンジンの単一インストールのみが必要です。このインストールはクラスターのいずれかのノードを介して実行でき、仮想ホスト名を指定する必要があります。

したがって、Information Server エンジン・ソフトウェアのバイナリー、アプリケーション・データのマウント・ポイント、およびディスク・パスはすべて、両方のサーバーで同一です。また、クラスター・ソフトウェアは、クライアントおよびその他の層が接続される仮想ホスト名を提供します。

アクティブ / パッシブとは対照的に、並列エンジン構成ファイルにはコンダクターの仮想ホスト名が含まれていますが、他のノードでステージが実行される 1 つ以上の物理ホスト名は含まれていません。*コンダクター*は、並列ジョブが起動されるサーバーです。

アクティブ・パッシブとは対照的なもう 1 つの点は、フェイルオーバー処理中に、ディスクのマウント・ポイントが既に両方のノードにマウントされているために変更されないままになることです。ただし、クラスター・フェイルオーバー・スクリプトには、変化する並列エンジン構成ファイルをすべて含め、残りの物理ホストのみが存在するようにする必要があります。

2.7.7 超並列処理 (MPP) トポロジー

Information Server 並列エンジンの超並列処理 (MPP) トポロジーの中で、各処理ノードには独自の CPU、ディスク、およびメモリーが割り当てられます。エンジン・ワークロードを処理する際、これらのハードウェア・リソースはノード間で共有されません。この方法は、*シェアード・ナッシング・アーキテクチャー*と呼ばれます。

図 2-16 に、シェアード・ナッシング、対称型マルチプロセッサー (SMP)、およびユニプロセッサーのアーキテクチャーを比較した相違点を示します。グリッド・アーキテクチャーも図に示されていますが、これは動的ワークロード管理を組み込んだ MPP の改良版です。グリッドについては、2.7.8『並列エンジン・グリッド・トポロジー』（44 ページ）で詳しく説明しています。

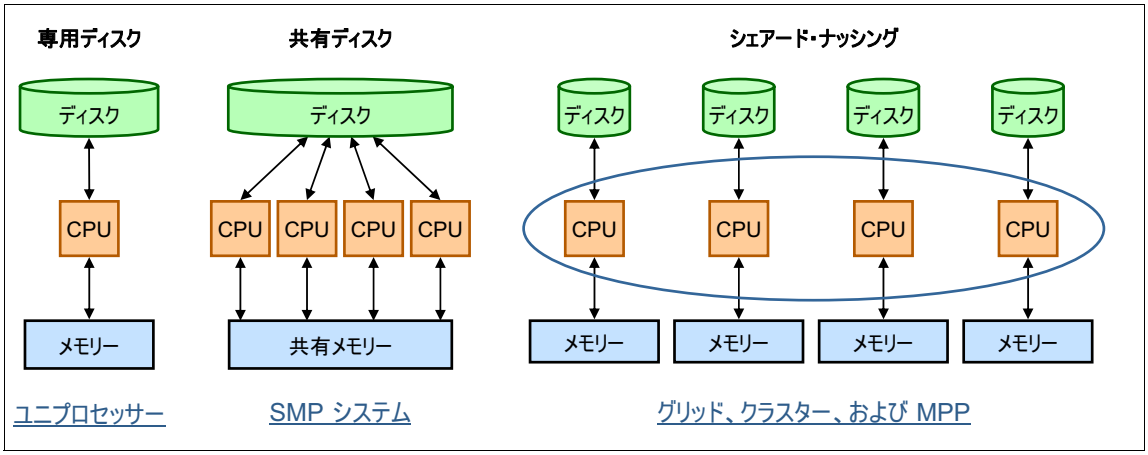


図 2-16　コンピューティング・アーキテクチャー

Information Server 並列エンジンの MPP アーキテクチャーの中では、高速プライベート・ネットワークを使用してノードをリンクすることをお勧めします。(『並列エンジン構成ファイル』（38 ページ）で紹介した) 並列エンジン・ジョブ構成ファイルは、fastname プロパティーを使用する高速ネットワーク上のノードのネットワーク名を記述します。図 2-17 (42 ページ) に、相互接続のために別の高速ネットワークを使用する 4 ノード構成の MPP を示します。この図で、_caa 拡張子は、高速ネットワーク・インターフェースのドメイン・ネーム・システム (DNS) 名を示しています。

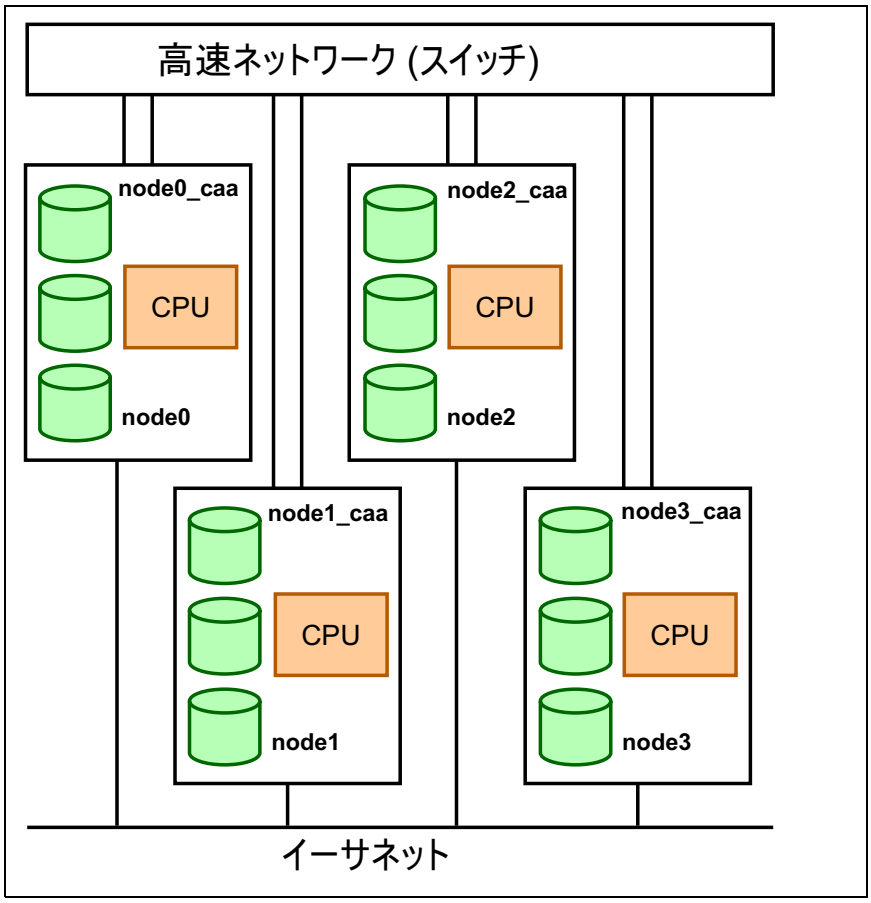


図 2-17　プライベート・ネットワークを使用する MPP

ノード間の認証に通常はセキュア・シェル (SSH) 接続が使用されますが、必要な場合はリモート・シェル (rsh) も使用できます。

この図では、明確にするためにその他の層を省略しています。

MPP 内で実行されていてジョブ・パーティションを必要とするリソースは、そのノード内から完全にアクセス可能でなければなりません。

MPP ソリューションを設計する際、以下の 2 つのタイプの処理ノードを検討してください。

- コンダクター・ノード (ヘッド・ノードとも呼ばれます)
- 処理ノード (計算ノードとも呼ばれます)

図 2-18 に示すように、並列エンジン・ジョブの実行時にシステム・プロセスの階層が作成されます。

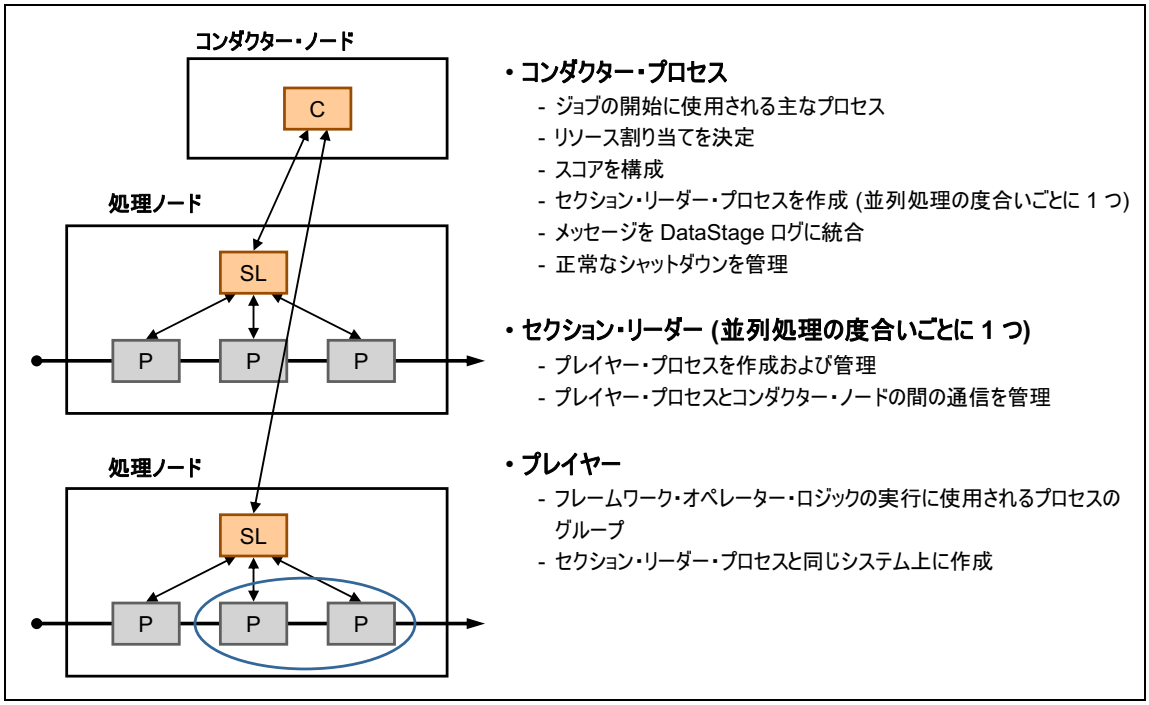


図 2-18　並列エンジン・プロセスの階層

コンダクター・プロセスは、ジョブの正常な作成とシャットダウン、ジョブを構成するプロセスの集合からのジョブ・ログ・メッセージの統合を実行します。プレイヤー・プロセスは、ジョブで処理を実行するステージであるため、これらのプロセスがサーバー負荷の大部分の原因となります。プロセス・ツリーは、ジョブ構成ファイルで指定されている情報に対応するサーバー内とサーバー間の両方で実行されるように設計されています。

並列エンジンのコンダクター (ヘッド) ノードで実行される処理がこれですべてである場合、このノードの負荷は軽くなります。MPP およびグリッド・アーキテクチャーでは、ヘッド (コンダクター) ノードのサイズを変更して、Information Server サービス層、リポジトリー層、または両方も実行できます。これにより、サービス層とリポジトリー層の密結合が確保され、サーバー・リソースに関する並列エンジンの処理ノードの競合は生じないため、この組み合わせは Information Server の層の要件に適しています。

図 2-19 に、ヘッド・ノードと計算ノードを示します。

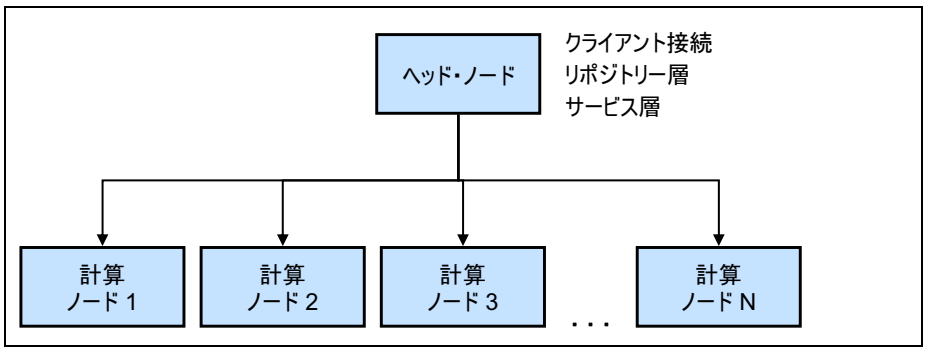


図 2-19　ヘッド・ノードと計算ノード

MPP エンジン層トポロジーのメリットと考慮事項

メリットを下記に示します。

- すべてのエンジン・ノードは、使用可能である間にデータ統合処理に使用できます。
- 単一サーバーの障害による使用可能な総サーバー・リソースへの影響は、対応して部分的なものです。

考慮事項を下記に示します。

- プライベート・ネットワークが推奨されます。
- Information Server エンジンのバイナリーを各ノードにコピーするか、すべてのノードにマウントされている共有ファイル・システムを使用して共有することができます。
- それぞれの DataStage ジョブ、または多数のジョブのサブシーケンスに使用される並列構成ファイルは注意して選択する必要があります。重要な作業として、ソリューションおよびその他のワークロードを実行できるように、すべてのサーバーが使用可能であるときにジョブが*通常*実行されるノードを検討します。ジョブとプロジェクトの数が増えると、ジョブ、ジョブ実行スケジュール、およびプロジェクトにマップされる構成ファイルの選択、管理、および維持は複雑になります。この問題を軽減する方法については、2.7.8『並列エンジン・グリッド・トポロジー』（44 ページ）で説明しています。

2.7.8 並列エンジン・グリッド・トポロジー

グリッド・エンジン・トポロジーは、2.7.7『超並列処理 (MPP) トポロジー』（41 ページ）で説明した MPP トポロジーの拡張版として考えられます。このセクションでは、相違点について説明します。

グリッド・トポロジーでは、リソース・マネージャー・コンポーネントを追加することで動的ワークロード管理機能が追加されます。

リソース・マネージャーにより、ジョブ並列エンジン構成ファイルの動的な生成が可能になります。そのため、この機能を使用すると、選択されたワークロード管理アルゴリズムに対応する使用可能なマシン・ノードにジョブを動的に割り当てることができます。ワークロード・アルゴリズムは、統合されているリソース・マネージャー・ソフトウェアの機能に基づいて構成することもできます。例えば、実行時のそのノードの負荷、ジョブの優先順位、時刻、またはこれらの決定事項やその他のカスタマイズ可能な決定事項の組み合わせに基づいて構成できます。

以下は、Information Server グリッド環境にデプロイされたリソース・マネージャーの例です。

- IBM Platform Load Sharing Facility (Platform LSF®)
- IBM Tivoli Workload Scheduler Load Leveler (Load leveler)
- Oracle Grid Engine (OGE (旧称 Sun Grid Engine SGE))

ワークロード・マネージャーの選択は、お客様の好みで決定できます。

グリッド環境の中で、ワークロード管理は、Information Server にリンクして、並列ジョブ構成ファイルで使用するノードを動的に生成します。

ワークロード・マネージャーは、稼働状態または停止状態のノードを追跡できるため、停止状態のノードはサービス停止の原因となりません。

リソース・マネージャー・コンポーネントには、各サーバー上で稼働するエージェントと、リソースの要求に対応するマスター・コーディネーターがありま

す。図 2-20 に示すように、マスター・コーディネーターは通常はヘッド・ノードに配置されます。

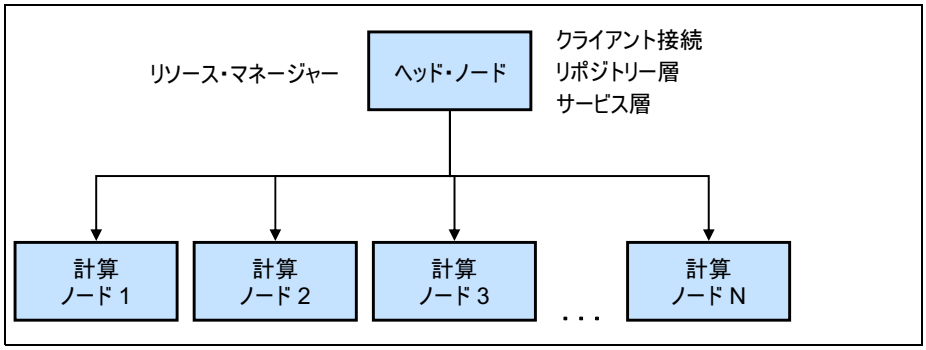


図 2-20　グリッドのヘッド・ノードと計算ノード

グリッド・トポロジーのメリットと考慮事項

メリットを下記に示します。

- エンジン層の大規模なスケーラビリティーを提供します。
- アイドル時間が長い部門別サーバーのサイロと比較して、大規模なインフラストラクチャー統合が可能です。
- カスタマイズ可能なワークロード管理を提供します。
- 共有サービスまたは仮想化プラットフォームを提供します。これにより、お客様のプロジェクトからプラットフォームが明確に分離されます。
- プロジェクト課金モデルを開発できる可能性を提供します。
- 複数の環境およびプロジェクトが計算ノードのプールを共有できます。
- MPP と比較して、管理と複雑さは比例して変化します。
- 部門別ではなく組織的な標準を推進します。
- 計算ノード・プールは本質的に高可用性です。計算ノードが失われても、完全なサービス停止は起こりません。
- サービスを停止することなく、計算ノードを追加できます。
- 複数の環境 (開発、テスト、疑似本番) は計算ノードのプールをワークロード管理と共有でき、プロジェクト・ライフサイクルのどの時点でも使用可能なハードウェア・リソースを最も有効に使用できます。

考慮事項を下記に示します。

- 内部ノード接続用の高速プライベート TCP/IP ネットワークが使用可能であることを確認してください。
- 並列エンジンの標準的手法と同様に、並列エンジン・ノード固有のスクラッチ領域には最も高速なディスクを使用します。例えば、予算が許せば、並列エンジンのソートおよびバッファー領域でのソリッド・ステート・ドライブの使用を検討します。
- ノード間で共有されるデータ・セット領域に高速ディスクを使用します。
- クラスター・ソフトウェアを使用して、ヘッド・ノードを計算ノードでクラスタリングし、ヘッド・ノードの高可用性を確保します。
- Information Server インストール・ディレクトリーを含む共有ストレージ領域に NAS または SAN ストレージを使用できます。SAN ストレージには本質的にサーバー間ファイル共有プロトコルは含まれていません。
- サーバー間ファイル共有プロトコルを本質的に含まない SAN ストレージの場合、データ・セットおよび共有データ領域にクラスター・ファイル・システム・テクノロジーが必要です。例として、IBM General Parallel File System (GPFS) または Red Hat Global File System (GFS) が挙げられます。

2.7.9 複数専用エンジン・トポロジー

図 2-21 に示すこのトポロジーは、各エンジンが独自のサーバーにインストールされている複数のエンジン・インストール環境で構成されます。ただし、すべてのインストール環境がリポジトリー層とサービス層を共有します。

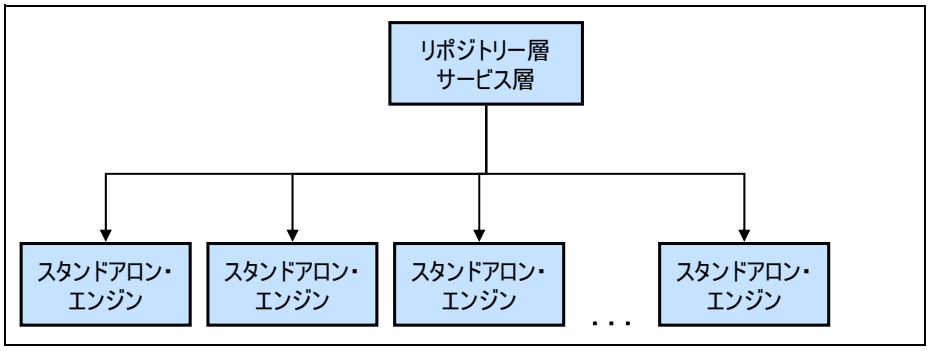


図 2-21　複数専用エンジン・トポロジー

この場合、コンダクター・プロセスを含むすべてのエンジン・プロセスは、関連エンジン・サーバーで実行されます。グリッド・トポロジーとは対照的に、共有されるストレージまたはリソース管理コンポーネントはありません。各エンジン・インストール環境には、ジョブ構成ファイルがあり、このファイル内でノード名はエンジン固有のサーバーを指示します。

このトポロジーを使用する主な理由は、単一リポジトリーおよび統合されたガバナンスとメタデータ・プログラムを使用できるようにする一方で、DataStage、QualityStage、および Information Analyzer のプロジェクトでサーバーへのエンジン・リソースをプログラムごとに完全に分離することです。*プログラム* という用語は、ここでは、他のサーバーから完全に分離され、独自のエンジン・インストール・サーバーで実行される一連のプロジェクト (DataStage、QualityStage、または両方) で構成されたビジネス・プログラムを指しています。そのため、メリットは、メタデータの共有ではなく、プログラム・エンジンの分離です。

このトポロジーのメリットと考慮事項

メリットは、並列エンジン・プロジェクトの完全なリソースの分離と、メタデータを共有できる機会です。

考慮事項を下記に示します。

- 複数のエンジン・サーバーにわたるワークロード管理は不可能です。
- それぞれのエンジン・サーバーのサイジングは、ピーク時のワークロードに対応して個別に行う必要があり、通常、システム全体で必要となるサーバー・サイジングの総容量は非常に大きくなります。
- エンジンのインストール環境の本質的な高可用性は存在しません。エンジン・サーバーの障害により、そのサーバー上のすべてのプロジェクトはサービスから除去されます。そのため、各エンジンのインストール環境をクラスタリングする必要があります。したがって、高可用性が必要な場合、インストール環境の複雑さは規模と比例せずに高まります。

2.7.10 Web サービス・エンジン

Information Server は、コンポーネントがリモート Web サービス・クライアントに応答できる Web サービス・デプロイメントをサポートします。その他のアーキテクチャーとは対照的に、Web サービス・デプロイメント内では、個々の DataStage ジョブが有効なサービス内で常に実行されるため、これらのジョブは着信する Web サービス要求に即時に応答できます。

Information Server Web サービス・トポロジーの高可用性とロード・バランシング機能の両方を提供するトポロジーの例を図 2-22 (47 ページ) に示します。ここで、DataStage (DS) および QualityStage (QS) のジョブと構成は、両方のエンジン層で完全に同じようにデプロイされます。着信要求は、ロード・バランシング・アルゴリズムに対応するいずれかのレグに経路指定されます。DataStage プロジェクトおよびジョブは、独立したインストール環境を持つ両方のサーバー・レグにデプロイされます。いずれかのレグでサーバーの障害が発生すると、すべての要求が残りのレグに送られることになるため、サービス停止はありません。ただし、サーバーの障害が発生している間、総処理容量は対応して減ります。

フロントエンドおよびサービス層のロード・バランシングについては、2.7.2『サービス層のクラスタリング』（28 ページ）で説明しています。

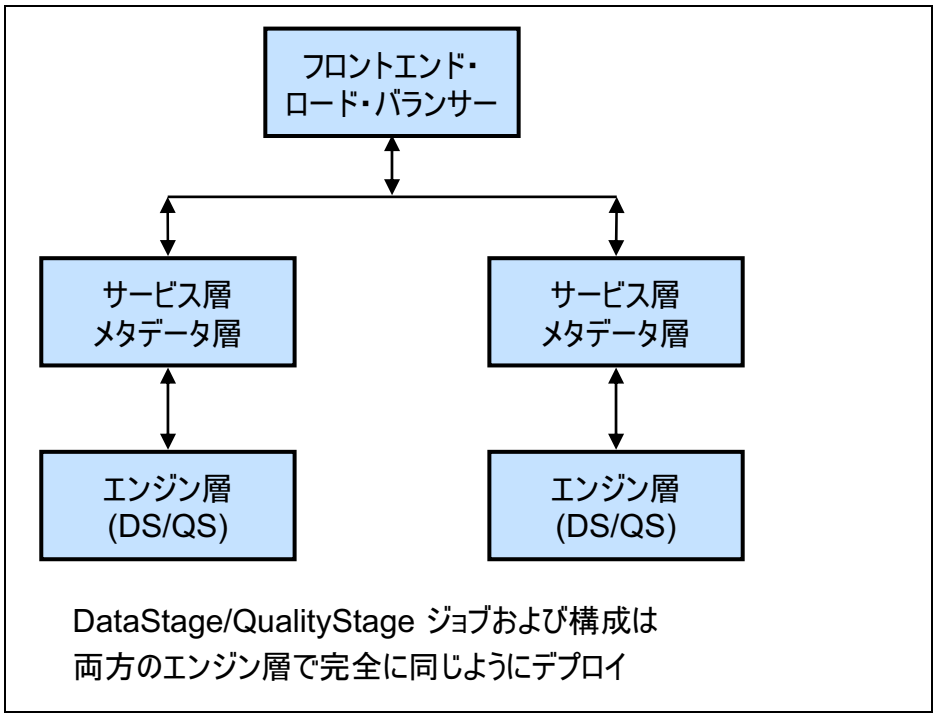


図 2-22　高可用性の Web サービス・デプロイメント

このトポロジーのメリットと考慮事項

メリットを下記に示します。

- 高可用性
- ロード・バランシング機能

考慮事項は、DataStage および QualityStage のプロジェクト、ジョブ、およびルールを両方のエンジン・インストール環境で同期してデプロイする必要があることです。

2.8 災害復旧

高可用性 (HA) を *災害復旧 (DR)* と区別することは重要です。本章のこれまでのセクションでは、単一コンポーネントの障害が原因で長期的なサービス停止が起こることはない高可用性トポロジーについて説明しました。

組織の災害としては、データ・センター全体が作動不能になる、つまりすべてのサーバーを使用できなくなる原因となる停電、火災、天災などが考えられます。この場合、そのデータ・センターにフェイルオーバー・ハードウェアがあっても助けになりません。

解決策は、組織が地理的に別の場所でデータ・センターの完全な冗長性を準備することです。

この問題に対する全社的なアプローチを実行する際、組織は、2 次データ・センターにストレージを複写するために組織の代わりにストレージ・システム・レベルで複製できる機能を持つ共通のストレージ・サブシステム (通常は SAN または NAS) を使用できます。

組織は、地理的フェイルオーバー機能を持つデータ・センター内に Information Server をデプロイできます。Information Server の 1 つの層が別の地理的位置にフェイルオーバーされる場合に、そのセンター内の Information Server のその他すべての層もその地理的位置にフェイルオーバーされることを考慮する必要があります。つまり、どちらかの場所で、スイートのインストール環境として稼働します。

Information Server では、バックアップから 2 次データ・センターへの SAN の複製または完了レベル 0 (ゼロ) のファイル・システム・リストア（フルリストア）のどちらを選択するかを検討します。SAN の複製は、リポジトリー層に関して 『HADR クラスター』（34 ページ）で説明した DB2 HADR と組み合わせることができます。

データ・センターのフェイルオーバーの後、Information Server をホストするようになった新しいサーバーに、さまざまな層をホストする元のサーバーと同じ DNS 名が付けられていることが重要です。Information Server リポジトリーには、維持される必要があるホスト名への参照があります。

SAN の複製のメリットと考慮事項

メリットは、データ・センターのフェイルオーバー時に、複製されたすべてのファイル・システムが最新状態であることです。

考慮事項は、データ・センター障害の正確な発生時点によっては、ファイル・システムの一部の側面が複数の層で同期されていない可能性があることです。結果として、データ・センターのフェイルオーバーの後で、手操作による介入が若干必要になる可能性があります。

レベル 0 リストア（フルリストア）のメリットと考慮事項

メリットは、バックアップに含まれるすべての層を完全に同期できることです。

考慮事項は、最後の正常なバックアップの時点からデータ・センターのフェイルオーバーの時点までの Information Server のデータ損失です。

2.9 本章のまとめ

本章では、多様なデプロイメント・トポロジーを、関連するメリットと考慮事項とともに説明しました。Information Server は優れた柔軟性を提供するため、組織のソリューション設計者は、選択したプログラムの目標や環境と組織のプラットフォームの要件と標準に適合する方法を選択できます。

デプロイメント・テクノロジーは、1 台のノートブックのインストールから、ワークロード管理をカスタマイズできる可用性の高い Enterprise Shared Service プラットフォームまで多岐にわたります。

3

メタデータ管理

情報ガバナンスの規則は、組織がデータの爆発的増加、動きの速い市場、上昇する不確実性のレベル、増大する規制要件の負担という課題への対処を改善する上で役立ちます。情報ガバナンスを実践すると、ユーザー、プロセス、およびテクノロジーの調整により、情報が理解され、高品質で信頼できるものであることが確保されます。

メタデータ管理は、情報ガバナンスの中心となる原理です。*情報ガバナンス*とは、組織が情報に関する知識を管理して、「情報に関して何を知っているか」などの質問に回答することができる能力を指します。

*メタデータ管理*とは、組織が「どのようにしたら、データに関する自分たちの理解内容を把握できるか」という質問に答えられるようにするためのツール、プロセス、および環境を指します。

本章は以下のトピックで構成されています。

- メタデータの定義
- メタデータのタイプ
- メタデータが重要である理由
- メタデータを表示する場所

3.1 メタデータの定義

メタデータの定義を検索すると、いくつかの定義が見つかります。定義は、メタデータが導入されているコンテキスト・ドメインによって異なります。例えば、ライブラリー管理のコンテキストにおけるメタデータは、ライブラリー資料のカタログ作成の側面を重視します。ライブラリーは、メタデータを使用して、ユーザー、ライブラリアン、および研究者がこれらの資料の特定、ソート、およびトレースで使用できる形で、ライブラリー・リソースおよびさまざまなタイプの資料を収集して記述します。研究者は、主題、名称、または作成者で資料を検索したり、同じ主題、同じ作成者、または同じジャーナル内の関連資料を検索したりすることができます。ライブラリー・リソースの使用率が数倍も高まると同時に、必要な作業は大幅に減ります。

不動産の分野では、メタデータは不動産リストに関する情報を収集します。仲介業者は、メタデータを使用して、リスト、履歴、および場所を記述する情報を収集します。買い手と売り手は、この情報を使用して比較可能なリストを調査することができます。

情報技術 (IT) の分野では、メタデータは、データおよびその他の IT 成果物に関連します。最も一般的なメタデータの定義は「データに関するデータ」です。ただし、コンテキストのないデータは無意味な一連の記号でしかないため、メタデータに関する議論はすぐに「データに関する情報、情報に関する情報」に立ち戻ります。関係とデータ・フローを抽出でき、検討と調査を行える形で大量の情報の集合が編成されていて、実用的な情報となっている場合、メタデータを「データに関する知識」と呼ぶことができます。

データ成果物のカタログを維持することは、ライブラリー内の資料のインベントリーの管理や不動産仲介業者の不動産リストの管理と似た目的を果たします。成果物の内容を理解して、その発信元をたどり、使用法をモニターして、ユーザーおよびステークホルダーにサービスを提供する必要があります。

情報処理の分野では、オブジェクトは、データ資産の電子レコードです。このようなデータ資産には、データ・ファイル、データベース内のデータ列、ビジネス・インテリジェンス (BI) レポート、またはデータ・ストア間でコンテンツを移動する変換ジョブが含まれます。これらのデータ資産は、企業がビジネス運営のために作成、保守、および使用するタイプのものです。これらのデータ資産に関する情報は、組織内の個々のユーザーにとって最も貴重なものです。データ・アナリストおよび開発者は、これらの資産の物理的特性を知る必要があります。

メタデータには、データ・エレメントの物理的特性を記述する技術的属性が含まれ、多くの場合は、意味、使用法、所有者、およびユーザーなどの意味情報と管理情報を提供する記述属性も含まれます。この情報には、組織のあらゆるレベルで多様な役割を持つユーザーによる幅広い用途があります。

3.2 メタデータのタイプ

メタデータには多くのソースがあり、多くのユーザーがオブジェクトに関する情報のさまざまな側面に利便性を見出しています。メタデータを内容と用途でさまざまに分類できます。IBM InfoSphere Information Server は、メタデータをビジネス、テクニカル、およびオペレーショナルのメタデータ・タイプに分類しています。それぞれのタイプは、内容とソースで区別されます。

3.2.1 ビジネス・メタデータ

*ビジネス・メタデータ*には、ビジネス用語とその定義、使用例、ビジネス・ルール・ポリシー、および制約が含まれます。これらは全体として、ビジネスの概念の意味体系と物理データ資産におけるそれらの具現化を定義します。

ビジネス・メタデータは、以下のタイプの質問に答えることによって、ビジネス・ユーザーとユーザー・コミュニティーのニーズに対応します。

- レポートに利益幅が示されているが、どのような意味があるのか
- 利益幅を計算するためのデータはどこから来たのか
- どのような計算によって利益幅が決定されたのか
- 誰 (どの*データ・スチュワード*) がこの用語を所有しているのか
- どのようなビジネス・ルールが利益幅に適用されたのか
- 他のどのレポートが利益幅を示しているのか

ビジネス・メタデータのユーザーは主にビジネス・ユーザーですが、誰でも物事の意味を理解するために使用できます。例えば、メタデータが使用された方法と時期や使用者と、データ資産の使用に適用されるポリシー、ルール、および制約事項が挙げられます。

ビジネス・メタデータのソースは主にビジネスです。対象分野の専門家 (SME) およびデータ・スチュワードがビジネス・メタデータのソースとなる場合があります。内部と外部のビジネス、運用、およびポリシーのマニュアルもビジネス・メタデータのソースとなる場合があります。

ビジネス・メタデータは、ビジネス・ユーザーが話して理解している言語で表現する必要があります。つまり、用語を詳しく説明して、定義が明確で一義的であることを確認しなければなりません。ビジネス・ルールの表現には、明白な言葉を使用して、省略語、数式、機能表現を用いる複雑で形式的な方法は用いません。

3.2.2 テクニカル・メタデータ

*テクニカル・メタデータ*は、データ資産の技術的な記述で構成されます。テクニカル・メタデータには、以下の記述が含まれます。

- スキーマ、テーブル、およびファイル・レイアウト
- ソースとターゲットのデータ・ストアの識別と物理的属性
- データ品質ルール
- ETL プロセス
- 変換ジョブ、ビジネス・ルール、その他のプロセスの正式な仕様

ユーザーとシステムがテクニカル・メタデータを使用します。アナリスト、開発者、管理者といった IT 技術スタッフは、業務を遂行するために日常的にテクニカル・メタデータを使用します。例えば、新規ジョブの要件を分析して仕様を作成したり、メソッドを開発してコードを作成したり、問題を診断して問題のフィックスを開発したりします。

データと、データを操作するプロセスの知識は、ビジネスにとって貴重なものです。テクニカル・メタデータは、以下の質問に回答する上で役立ちます。

- どのようなデータベースが存在しているか

- データベース内にどのようなスキーマ、テーブル、ビュー、および列があるか
- 表のキーは何か
- ジョブは表から何を読み取るのか
- 列に Null 値を含めることはできるか
- ジョブは表に何を書き込むのか
- データ・ストリーム内のフラットな順次ファイルは何か
- データはどのように処理されるのか
- どのような有効値が列で許可されるか

テクニカル・メタデータのソースは通常、データ資産そのものです。データベースは、データベース管理システムが保管データを操作するために必要とするすべての情報が含まれるデータ・カタログを維持します。BI ツールは、データを操作して望ましい形式で表示するためのデータ・モデル、関数、仕様、および命令を維持します。

データ統合ツールは、データのソース、ターゲット、およびフローの詳細、データに適用される変換とパラメーター、および指定された機能を実行するために必要なその他すべての詳細を維持します。さまざまなツールがタスクを実行するために使用するこのような情報はすべて、テクニカル・メタデータです。テクニカル・ユーザーはテクニカル・メタデータを作成および使用して、通常は個々のツールに保管します。テクニカル・ユーザーは、多くの場合、データ資産に注釈を付けたり、設計したり、文書をモデル化したりすることによって、手動でテクニカル・メタデータを改良します。

3.2.3 オペレーショナル・メタデータ

*オペレーショナル・メタデータ*は、アプリケーションまたはジョブの実行に関する情報で構成されます。このような情報には、時刻と日付、処理および拒否されたレコードの数、ジョブ実行の過程で生成されたプロセス、プロセッサー、およびサーバーに関する統計が含まれます。

オペレーショナル・メタデータは、以下の質問に回答するために使用されます。

- ジョブが最後に実行された日時はいつか
- 処理されたレコードの数はいくつか
- 拒否されたレコードの数はいくつか
- ジョブを処理するのにかかった時間はどれくらいか

オペレーショナル・メタデータのユーザーは、主に、さまざまなプロセスの実行パフォーマンスをモニターする運用担当者です。IT およびビジネスの管理者は、既存のリソースへのアプリケーションの追加の検討やマシン・リソースの追加に関する意思決定の際に、システム全体のパフォーマンスとスループットを知る必要があります。ビジネス・ユーザーは、さまざまなビジネス上の業務で使用するデータの即時性に関心を持つこともあります。そのため、最後にジョブが実行された時刻やデータ・ストアが更新された時刻を探します。ビジネス・ユーザーは、特定のジョブが実行された日時やデータ・ストアが更新された日時を調べることによって、回答を得ることができます。

3.2.4 メタデータの使用法の分類

本書の目的のために、ここではメタデータの 2 つの使用法の分類を明らかにします。これらのメタデータのサブタイプは、メタデータが組織によって表示および使用される時期と場所によって区別されます。メタデータの表示方法と使用目的を判別すると、組織がそれぞれの環境を判別する上で役立ちます。

デザイン・メタデータ

*デザイン・メタデータ*は、ビジネス・メタデータとテクニカル・メタデータの両方で構成され、統合プロジェクトの開発段階で使用されます。デザイン・

メタデータは、Information Server 内部で協調的に使用され、複数の統合タスクで再使用されます。デザイン・メタデータは、Information Server コンポーネントによって作成され、以下のようなアイテムで構成されます。

- 表分析結果
- データベース表定義
- ビジネス用語
- マッピング仕様

デザイン・メタデータは、データをソースからターゲットに移動するテクニカル・ユーザー、または統合プロジェクトの範囲内のビジネス・ユーザーが使用できます。

レポート・メタデータ

*レポート・メタデータ*は、ビジネス、テクニカル、およびオペレーショナルの 3 つのすべてのメタデータ・タイプで構成されます。このメタデータは、組織内のさまざまなデータ・ストア間のデータの移動と関係を理解するために使用されます。レポート・メタデータには、Information Server コンポーネントによって作成されたメタデータや外部ソースからインポートされるメタデータも含まれ、以下のようなアイテムが含まれます。

- ETL ジョブおよび実行結果
- データ・ソース
- ビジネス用語
- 外部のプロセスおよびアプリケーション
- BI レポート

レポート・メタデータは、データ・ガバナンスのイニシアチブの中で使用でき、データの正確性と妥当性における信頼性を構築する上で役立ちます。

3.3 メタデータが重要である理由

メタデータ管理とは、データ・オブジェクトに関する情報をカタログに登録することだけではありません。関連するメタデータを識別して、それらの品質を高め、即時性を維持して、再使用を促進できるようにメタデータを保持することも含まれます。メタデータ管理は、組織が「どのようにしたら、データに関する自分たちの理解内容を把握できるか」という質問に答えられるようにするツール、プロセス、および環境を提供します。データ・オブジェクトと、それらの意味、物理的な場所、特性、および使用法に関する情報を簡単かつ便利な方法で特定および取得できる能力は、強力で、企業にとって有益です。この能力は、組織がリスクに対処して、規制要件に対応し、IT の生産性を改善する能力を高めます。

3.3.1 信頼できる情報によるリスク回避

組織は、数多くのタイプのリスクに直面します。ほんの数例を挙げると、市場、運用、規制に関する脅威があります。情報と知識は、組織がリスクを緩和できる手段です。信頼できる情報に基づいて計画し、意思決定を下すことができても、リスクはなくなりません。ただし、この能力があれば、企業は、直面しているリスクをより正確に評価して、適宜に行動することができます。組織内のデータとシステムの量が増えるにつれて、データ統合は複雑化し、それ故に信頼できる情報の重要性が高くなります。情報ガバナンスのすべての要因は、意思決定のために報告および使用する情報が信頼できるという前提に依存します。メタデータ管理は、企業が必要とする信頼性の手段となります。データ・リネージュと影響分析を通じて、企業は、計画または意思決定のモデルに使用されるデータの正確性、完全性、および即時性を把握できます。

3.3.2 規制準拠

組織は、かつてないほど規制要件に制約されるようになっています。特定のビジネス領域における活動について定期的な報告書を提出することが義務づけられています。ある種の規制の指令は、データの扱いを重視しており、さまざまなタイプのデータのセキュリティー、プライバシー、および保存の要件を規定しています。組織は、頻繁に監査の対象となります。その場合、それらの要件に準拠していることを示し、実証する必要があります。スチュワードシップ、データ・リネージュ、および影響分析のサービスを提供する統一プラットフォームで実施されるメタデータ管理は、組織が、報告対象のデータが真実であることを立証および実証することができる最善の保証です。また、データが正確であり、規制に従って扱われていることを立証および実証します。

3.3.3 IT の生産性

多くの組織は、新しいサービスまたは報告要件の支援を依頼された際、同じデータおよびプロセスを分析し、再分析する反復的な IT の業務に難渋しています。データの量は豊富です。その多くは既存のシステムとデータ・ストアから発生しますが、それらについての文書は存在しません。文書が存在したとしても、それらのシステムとデータ・ストアの多くの変更と更新が反映されていません。多くの場合、この貴重な知識は、システムを最初に開発した個人が持っていますが、その人が退社したり、他の部門の職務や業務に異動している場合があります。

この文書の欠如は、新しいタスク、そのプロパティー、および依存関係に必要なエレメントを識別するためのリバース・エンジニアリング、調査、および分析という手間のかかる作業につながります。開発者、アナリスト、その他のステークホルダーがアクセスできるリポジトリーにメタデータが収集および維持されていると、この労力を部分的に回避できます。

3.3.4 メタデータを管理するための要件

メタデータ管理の要として、ビジネスと IT の間の徹底したパートナーシップが必要です。その際、共同の所有権と管理責任において、両方のコミュニティーが時間と労力の面で貢献し、最終成果物の外観と機能について情報を提案します。

メタデータ管理のイニシアチブは、通常、さらに範囲が広い情報ガバナンス・プログラムの領域の中に組み込まれます。メタデータ管理は、実現手法であり、報告、計画、および意思決定のために信頼できる情報を提供することによって日常的な業務における管理をサポートします。メタデータ管理は、ユーザーが情報の意味を表明して、その正確性、完全性、および即時性を判別する上で役立ちます。

長年にわたり、情報専門家と企業経営者は、データは貴重な資産であり、慎重に扱う必要があることを認識してきました。大半の組織は情報資産の価値を文書化していませんが、利用可能な情報から価値を引き出せること、そして情報を適切に扱って保護しないと損害を被る可能性があることを認識しています。

情報ガバナンスの規則は、健全な企業を維持するための第一条件となっています。現在、法律または規制により、多くの企業および政府機関は、データの保全性、セキュリティー、およびプライバシーを規定レベルで維持して、これらの要件への準拠を提示できることを義務づけられています。その上、組織が高品質の情報を保証して情報に関する知識を維持する能力は、今日の市場競争で成功を収める上で非常に有益です。

数十年間にわたり、情報システムは進化して、相互に情報を供給するアプリケーションとデータ・ストアの複雑な網構造が形成されています。トランザクション・データ・システムでは貨幣または物資の取引が行われ、資金が口座間

で送金され、旅行が予約され、購入は記録され、輸送指示書が発行されています。これらのトランザクションは、企業がパターンを抽出して傾向を明らかにするために収集して調査する貴重な情報のリポジトリーを表しています。

企業は、このデータを理解し、管理者やその他のナレッジ・ワーカーが計画と意思決定のために使用できる方法でデータを提示するために、BI アプリケーションを使用します。企業は、パイプ、チャネル、およびアプリケーションで構成される情報サプライ・チェーンを形成します。IT では、これらのデータの経路に加えて、データのクレンジング、統合、および調整のために多様なツールとユーティリティーを使用します。これらのツールは、さまざまな情報のニーズに対応して設計されたマスター・データ・リポジトリー、データウェアハウス、およびデータマートにデータを取り込みます。

組織は、その時々で多様な情報処理イニシアチブを導入します。新しいオファリング、製品、およびサービスをサポートするためにアプリケーションをアップグレードする場合でも、コンプライアンスの報告要件に対応する場合でも、新しい分析や意思決定の規則を採用する場合でも、IT 部門は、大量の情報をシステム間で移動して、そのフォーマットや表現を変更し、多忙に作業します。

現在の情報環境では、このような作業には多くのリスクが伴います。組織が法的責任と規制上の責任を果たすことができず、リスクを被る可能性があるのは、以下の問題に起因します。

- 完全で正確な仕様を取得できない
- データ・エレメントの意味と使用法の理解が欠如している
- データ・エレメントに適用される可能性がある制限と制約の理解が欠如している

従来型の情報管理では、知識は、個々のシステムに格納されているか、その分野の知識を持つ個人のみが持っているか、文書化されている場合は個人の端末や引き出しにしまい込まれていました。このような文書をすべて公共の領域に保管するための共有ドライブや指示は若干の緩和策となりますが、それらはすぐに時代遅れになります。企業がアーキテクチャー設計、データ・ディクショナリー、システム仕様、その他の文書の中にデータを配置する場合、これらのリポジトリーは、簡単に検索および調査するためには役立ちません。情報が 1 つの場所に集められて大規模なユーザー・コミュニティーが利用できる状態であっても、正確な文書を識別して、その即時性と関連性を判別し、情報を取得するには多大な労力が必要となります。

このスタイルの運用に伴うコストとリスクは高くつき、ずっと以前から組織はこの問題を認識していました。長年にわたって導入されてきたシステム開発方法論により、ソフトウェア開発プロセスの簡素化および合理化は大幅に前進しています。ただし、導入後のプロセスで生成される知識の保存のサポートについては、あまり多くの成果はありません。

メタデータ管理は、情報サプライ・チェーンに関する知識を管理する手法です。多くの人がメタデータを「データに関するデータ」と呼んでいますが、実際には、企業は単なるデータや情報を超えるものを扱っています。メタデータは、豊富な知識構造です。この構造は、用語またはデータ資産の意味、他の資産との関係、適用される可能性があるルールを収集して、品質、ポリシー、およびその用途を規定する規制を判別します。

メタデータ管理は、組織が今日の動きの速い世界で直面する多くの課題に対応します。取引は瞬く間に行われ、このスピードに対応した意思決定が必要です。この世界において、信頼できる情報は非常に貴重なものであり、データ・フローをトレースおよび追跡してデータに関連する情報を利用できることが決定的に重要です。

3.4 Information Server のメタデータ

Information Server は、内部および外部の両方で生成される多様なメタデータ資産を管理して、企業全体で情報を把握できるようにします。メタデータ管理の詳細な説明については、「*IBM InfoSphere Information Server によるメタデータ管理*」(SG88-4071) を参照してください。

3.4.1 Information Server 内部のメタデータ

すべての Information Server 製品モジュールは自動的にメタデータを生成するとともに使用します。生成されるメタデータは、InfoSphere Information Server リポジトリーで保持されます。この動作は、開発、テストまたは QA、本番、および企業にあるその他の環境など、すべての InfoSphere Information Server インスタンスに当てはまります。

内部で生成されるメタデータは、Information Server 共有リポジトリーで保持されます。共有リポジトリー内の資産は、Information Server 内のすべてのコンポーネントで使用可能かつ再使用可能です。例えば、Information Analyzer での分析時にインポートされる表定義は、即時に DataStage または Metadata Workbench で再使用できます。

3.4.2 Information Server 外部のメタデータ

Information Server には、その他のツールによって作成されるメタデータのインポート、品質強化、および維持のための機能があります。この外部メタデータは、最初に識別および収集された後で、共有リポジトリーにロードされる必要があります。このロードは、1 ページの第 1 章、『Information Server の概要』で説明したように、Import Export Manager または InfoSphere Metadata Asset Manager アプリケーションによって実行できます。

外部メタデータは、共有リポジトリーにロードされた後、Information Server 内のすべてのコンポーネントで使用可能になります。例えば、モデリング・ツールによって作成された表定義を Information Analyzer で使用できます。ビジネス・インテリジェンス・レポートを Metadata Workbench で使用して、データ・リネージュ・レポート内で表示できます。

3.4.3 メタデータのライフサイクル

ソフトウェア開発ライフサイクル (SDLC) 内のアプリケーション環境の間でマイグレーションするための標準的なアプローチでは、資産を移動するための詳細に定義されたステップに従います。ただし、Information Server 環境の中のメタデータは別のアプローチに従うことがあります。開発環境の中で定義されたメタデータは、本番などの他の環境に移動するには不要または不適切である場合があります。例えば、開発においてデータ・リネージュを作成するために使用されたオペレーショナル・メタデータは、本番には関係ありません。方法論とメタデータのライフサイクルについては、63 ページの第 4 章、『Information Server 資産のライフサイクル管理』で詳しく説明しています。

3.5 Information Server のメタデータを表示する場所

このセクションでは、専用のメタデータ環境を持つことの可能性について説明します。この専用の環境の目的は、組織内に存在していて Information Server に保管されているさまざまなタイプのメタデータにすべてのステークホルダーが常時アクセスできるようにすることです。

3.5.1 デプロイメントのためのビジネス要件

ビジネス要件に応じて、Information Server 製品モジュールは、環境の一部または全体にデプロイされます。例えば、DataStage は、さまざまな環境でプロモートするための開発サイクルの一部であるため、常にすべての環境にデプロイされます。Metadata Workbench は、開発環境ではインストール (使用) する必要はありませんが、本番環境にはインストールする必要がある場合があります。

専用の環境では、メタデータが作成および保持される環境に関係なく、関連するメタデータは、Information Server のこの別個のインスタンスにコピーされます。これは、メタデータを蓄積するための専用のインスタンスです。この専用の環境は、すべてのステークホルダーが必要なメタデータにアクセスして表示するための場所となります。

例えば、ビジネス・アナリストは、結果として生成されるデータが適切な形式で正確であることを求めます。データ・スチュワードや場合によっては CEO は、すべてのデータ資産が検証可能であり、存在する場合は規制上の制約に準拠していることを求めます。ETL 開発者は、ジョブを実行するためのアプリケーションを効率的かつ正確に開発する方法を知る必要があります。これらのすべての例では、Information Server への別々のアクセスが必要になります。

組織によっては、正当な理由から、本番サーバーへの不特定アクセスに反対します。その一方で、やはり正当な理由から、生成されるデータの信頼性を得るために本番データへのアクセスを要求する組織もあります。この環境は、さまざまな企業で実装されています。

検討すべき最も重大な意思決定のポイントが分かっている場合、企業の問題に合致したポリシーを設定できます。まず、以下の例のような Information Server が実行するタスクを重点的に検討してください。

- データの移動
- データのクレンジングまたは変換
- データの分析
- ビジネス定義の保管
- データ資産への所有者の割り当て

次に、この環境の開発者と運用スタッフ以外も含むユーザーを検討します。その他多くのユーザー・グループが何らかの方法での Information Server へのアクセスを必要としています。すべてのユーザーに有効な理由があることを考慮して、全ユーザーに本番システムへのアクセス権を付与するでしょうか。答えは、どのようにして各グループのニーズを評価し、それらのニーズをシステム・パフォーマンスと比較検討するかによって異なります。

基礎となる前提は、誰もが本番システムのパフォーマンスに影響を与えてはならないということです。例えば、夜間にのみデータを処理していて、負荷は段階的に増加し、実行は比較的短時間であると想定します。その場合、日中に Metadata Workbench を使用してそのシステム上のオペレーショナル・メタデータにアクセスできるようにすることは妥当でしょうか。繰り返しますが、答えは状況によって異なります。その他のオプションも検討してください。さらに多くのハードウェアとソフトウェアが必要になりますが、お客様が求める答えとなる可能性があります。

3.5.2 メタデータ環境

メタデータの使用と共有に関する組織の要件とニーズはそれぞれ異なります。実装する、または最初に導入する環境のタイプを決定する際、組織は、達成しようとしているビジネス目標を検討する必要があります。実装する環境を選択する際に考慮すべき主なビジネス目標は、IT の生産性と情報ガバナンスです。例えば、コラボレーションの強化による生産性向上を目標としていて、情報ガバナンスの優先順位付けはまだ行われていない場合があります。このような場合、コラボレーションを強化する環境を選択すべきです。組織内で情報ガバナンスの優先順位付けが行われるようになった場合、コラボレーションとメタデータ・レポーティングを向上させる環境を実装できます。

本書では、Information Server を使用する場合に考慮すべきメタデータ環境をシンプル、ユニファイド、レポーティング、およびガバナンスという 4 つのタイプに分類します。これらのメタデータ環境については、137 ページの第 5 章、『メタデータのデプロイメント・アーキテクチャー』で詳しく説明しています。

シンプル

シンプル・メタデータ環境は、Information Server プラットフォーム内にインストールされているコンポーネントとユーザー・コラボレーションの量によって定義されます。シンプル・メタデータ環境では、別々の業務を遂行するユーザー間のコラボレーションはごくわずかです。また、企業のメタデータ・レポーティング要件も重大ではない場合があります。例えば、InfoSphere Information Analyzer のみがインストールされている場合、ユーザー間のコラボレーションは必要ありません。また、DataStage のみがインストールされている場合、全社規模のメタデータ・レポーティングは必要ありません。

シンプル・メタデータ環境の考慮事項には、以下の質問が含まれます。

- どの Information Server ツールがインストールされているか
- ユーザーは業務を遂行するためにメタデータを共有するか
- メタデータ・レポーティングに関連して負荷が高くなることが予期されるか

ユニファイド

ユニファイド・メタデータ環境も、Information Server プラットフォーム内にインストールされているコンポーネントとユーザー・コラボレーションの量によって定義されます。ユニファイド・メタデータ環境では、類似した業務や関連業務を遂行するユーザー間で大量のコラボレーションが行われています。例として、InfoSphere Business Glossary に取り込まれたビジネス定義およびデータ分析や、InfoSphere FastTrack で作成されるマッピング仕様が挙げられます。オペレーショナル・メタデータも多くのユーザーに必要とされます。

ユニファイド・メタデータ環境の考慮事項には、以下の質問が含まれます。

- 本番環境と似た外観でありながら、本番環境に影響を与えないようにするには、どのようにしてこのインフラストラクチャーを作成するか
- メタデータまたはガバナンスのニーズに対応するために本番環境の一部を複製する場合はどうなるか

レポーティング

レポーティング・メタデータ環境は、協調的な開発環境からのメタデータを本番環境からのオペレーショナル・メタデータと結合しますが、本番環境に影響を与えないようにハードウェアの点では分離されています。Information Server のメタデータに基づくレポートには、独自のレポーティング環境が必要となることがあります。

レポーティング・メタデータ環境の考慮事項には、以下の質問が含まれます。

- 企業のメタデータ・レポーティングのニーズは本番環境に影響を与えているか
- 企業に大容量の複雑な辞書があるか

ガバナンス

ガバナンス・メタデータ環境は、DataStage および QualityStage がデプロイされていない場合に Information Server のレポーティング機能とガバナンス機能を使用します。ガバナンス環境は、実行されるメタデータ・レポーティングの量という点でレポーティング環境に似ています。

ガバナンス・メタデータ環境の考慮事項には、以下の質問が含まれます。

- どの Information Server ツールがインストールされているか
- 企業でメタデータ・レポーティングが必要であるか

4

Information Server 資産のライフサイクル管理

本章では、IBM InfoSphere Information Server Manager および外部のソース・コード管理ツールを使用した IBM InfoSphere DataStage および QualityStage アプリケーションのソース・コード管理を含むライフサイクル管理を実装する方法について説明します。また、この章では、IBM InfoSphere Information Analyzer、IBM InfoSphere Business Glossary、IBM InfoSphere Metadata Workbench、および関連するメタデータ資産のライフサイクル管理についても説明しています。

4.1 資産の概要

資産をやりとりするための **istool** コマンド・ライン・インターフェースを使用して、異なる InfoSphere Information Server インストール済み環境のメタデータ・リポジトリー間で個々の資産や大容量の資産のグループを移動できます。例えば、開発環境からテスト環境、本番環境、またはソース管理環境に資産を移動できます。大容量の資産のグループの日常的なバックアップや移動を容易に行えるように、資産交換コマンドをスクリプトとして作成できます。インポートまたはエクスポートのプロセスの速度は、プロジェクトの規模によって異なります。

表 4-1 に、資産交換を使用することによって移動できる資産をリストします。

Information Server 内部で使用できる製品のライフサイクル管理は、スイート内の各製品によって異なります。例えば、一般的に、DataStage および QualityStage 製品は開発環境、テスト環境、および本番環境にデプロイされますが、Information Analyzer 製品は開発環境と本番環境のみにデプロイされる場合があります。特定製品に適用される方法論は、本章の中のそれぞれのセクションで説明されています。

章のタイトルにある *資産* という用語は、関連する DataStage および QualityStage、または Information Analyzer、Business Glossary、または Metadata Workbench の資産の集合を指しています。これらの資産の詳細については、本章の中のそれぞれのセクションで説明しています。

表 4-1 資産交換を使用して移動できる資産

カテゴリー	資産
IBM InfoSphere Business Glossary	► カテゴリー ► 用語
IBM InfoSphere FastTrack	► マッピング・コンポーネント ► マッピング構成 ► マッピング仕様 ► プロジェクト・テンプレート ► プロジェクト ► プロジェクトに関連付けられているロールの割り当て、レポート、共通メタデータ、グロッサリー資産、および IBM InfoSphere DataStage と QualityStage の資産
IBM InfoSphere Information Analyzer	► プロジェクト (分析結果、データ・ルール、およびデータ・クラスなど、プロジェクトに関連付けられている資産を含む) ► すべてのデータ・クラス (関連付けられているプロジェクトに関係ありません) ► プロジェクトに関連付けられているレポートおよび共通メタデータ

カテゴリー	資産
IBM InfoSphere DataStage and QualityStage	► カスタム・フォルダー (プロジェクト・ディレクトリー内の外部ファイル) ► データ接続 ► データ・エレメント ► データ品質仕様 (ルックアップ・テーブル、ルール・セット、およびマッチング仕様) ► IBM IMS データベース ► IMS ビュー・セット ► ジョブ (メインフレーム、パラレル、シーケンス、およびサーバー) ► ジョブ実行可能ファイル ► マシン・プロファイル ► パラメーター・セット ► ルーチン (メインフレーム、パラレル、およびサーバー) ► 共有コンテナー (パラレルおよびサーバー) ► ステージ・タイプ ► 表定義 ► 変換 ► 表定義に関連付けられている共有表および関連する共通メタデータ資産
共通メタデータ資産	ビジネス・インテリジェンス (BI) 資産 : ► BI モデル ► BI コレクション ► キューブ ► BI レポート ► BI レポート照会 実装されているデータ・リソース : ► ホスト・コンピューター ► データベース ► データベース・スキーマ ► ストアード・プロシージャー ► データベース表 ► データ・ファイル ► データ・ファイル構造 ► データ接続 ► データ・アイテム定義 論理データ・モデル資産 : ► 論理データ・モデル ► 論理エンティティー ► 論理関係 ► エンティティー一般化階層 ► 論理領域 ► サブジェクト・エリア ► ダイアグラム 物理データ・モデル資産 : ► 物理データ・モデル ► 設計表 ► 設計ストアード・プロシージャー ► 物理領域 各種の資産 ► データ接続 ► カスタム属性 ► 連絡先ライブラリー
レポーティング資産	► レポート ► レポート結果

カテゴリー	資産
セキュリティー資産	► ユーザー (ロール、資格情報、資格情報マッピングの有無は関係ありません) ► グループ (ロールの有無は関係ありません)

DataStage および QualityStage アプリケーションのライフサイクル管理

本章のこのセクションでは、IBM Information Server Manager のほか、外部ソース制御システムとして IBM Rational ClearCase® ソース・コード管理システムを使用する DataStage および QualityStage の資産のライフサイクル管理方法について説明します。InfoSphere Information Server Manager は、ジョブおよび関連する資産を管理する上で役立つデプロイメント・ツールです。標準的な環境では、複数のシステムでデータ変換ソリューションを実装して、これらのシステム間での資産のプロモーションを制御します。環境には、少なくとも開発、テスト、および本番のシステムがあります。デプロイメント・ツールは Windows コンピューターにインストールされ、同じマネージャーからすべてのシステムに接続できます。

このセクションの文脈における*資産*という単語は、関連する DataStage および QualityStage の資産の集合を指しています。さらに、この文脈における資産は、DataStage プロジェクトに属するジョブ、共有コンテナー、表定義なども指しています。

4.2 ソース制御およびデプロイメントの概要

このセクションの主な目的は、Information Server の一部としてサポートするソース・コード管理とデプロイメントについて紹介することです。ここでは、ソース・コード管理統合の応用のパターンを推奨することにより、Information Server 資料における説明を拡大しています。このパターンは、これらのアプリケーションの開発とデプロイメントを最も迅速かつ効率的に行うためにテクノロジーを正しく使用する上で役立ちます。ソース・コード管理とデプロイメントについては、Microsoft Visual Studio または Eclipse などの統合開発環境 (IDE) を使用する場合に一般的な従来型のソフトウェア開発環境の文脈の中で説明しています。

次に、Information Server との対比を示しています。どちらのアプローチも*ソフトウェア主導型*ですが、開発とデプロイメントは根本的に異なります。特に、Information Server は、すべてのアプリケーション設計資産が保管される Information Server リポジトリー層を提供する点で独特であるためです。すべての開発作業とデプロイメント作業は常に Information Server 内で行われます。

後続のセクションでは、ソース・コード管理ツール (IBM Rational Team Concert、ClearCase、または CVS など) を Information Server と統合する方法、デプロイする方法、および効果的なソース・コード管理とデプロイメントをサポートするために DataStage プロジェクトを構造化する方法の実用的なアドバイスを詳しく説明します。本章では、外部ソース管理システムとして IBM Rational ClearCase ソース・コード管理システムを用いて、Information Server (IS) Manager ツールでの DataStage および QualityStage の資産のライフサイクル管理を説明しています。

先に進む前に、ここで説明する内容の「全体像」としての特徴を紹介することが重要です。図 4-1 を参照してください (図の中の「DS」は DataStage を指しています)。

この全体像には、次の主な 2 つの側面があります。

► パッケージのビルドとデプロイメント
► ソース・コード管理とバージョン管理

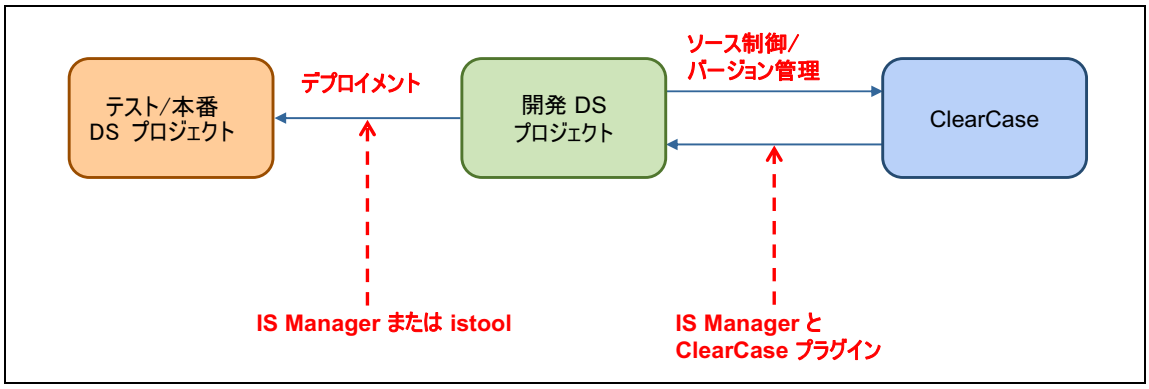


図 4-1　デプロイメントとソース・コード管理およびバージョン管理の全体像

デプロイメントには、具体的に IS Manager または **istool** コマンド・ライン・ユーティリティーのいずれかで対応します。ソース・コード管理とバージョン管理には、IS Manager へのプラグインとしてサード・パーティー・ツール (ClearCase など) を統合することによって対応します。

4.3 従来型のソフトウェア開発におけるソース管理およびデプロイメント

従来型のソフトウェア開発とは、一般に Microsoft Visual Studio または Eclipse などのスタンドアロン IDE が関与することを意味します。図 4-2 に、このタイプの環境を示します。

このような IDE は、より簡単かつ効率的に開発できるように、デバッグ、コンパイル、およびビルドなどのツールと便利な手段を提供します。

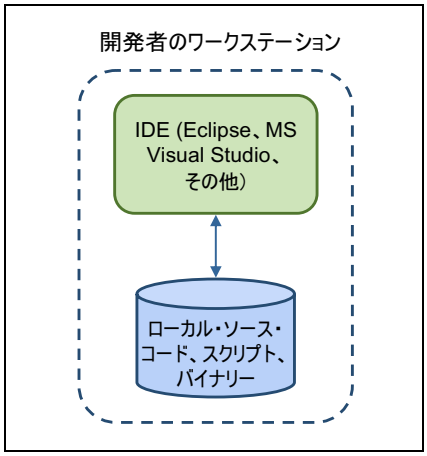


図 4-2　1 台の開発者のワークステーション内部での従来型のソフトウェア開発

開発者が 1 名の環境では、他の開発者の作業による更新およびビルドとの調整に関する懸案事項はありません。ソース・ファイルとアセンブリーのローカル・コピーは、簡単に維持および管理できます。唯一の課題は、開発に関して自制を課すことです。

図 4-3 (68 ページ) に示すように、複数の開発者の間で作業を調整しなければならないときに懸案事項が発生します。例えば、ソース・ファイルとアセンブリーを移動する方法、複数の個人の間で作業の同期を維持する方法などです。IDE は、引き続き、個人からのソース・ファイルのローカル・コピーを使用するローカル・ワークスペースを提供します。ただし、IDE は、内部でのソース・コード管理の機構は提供しません。

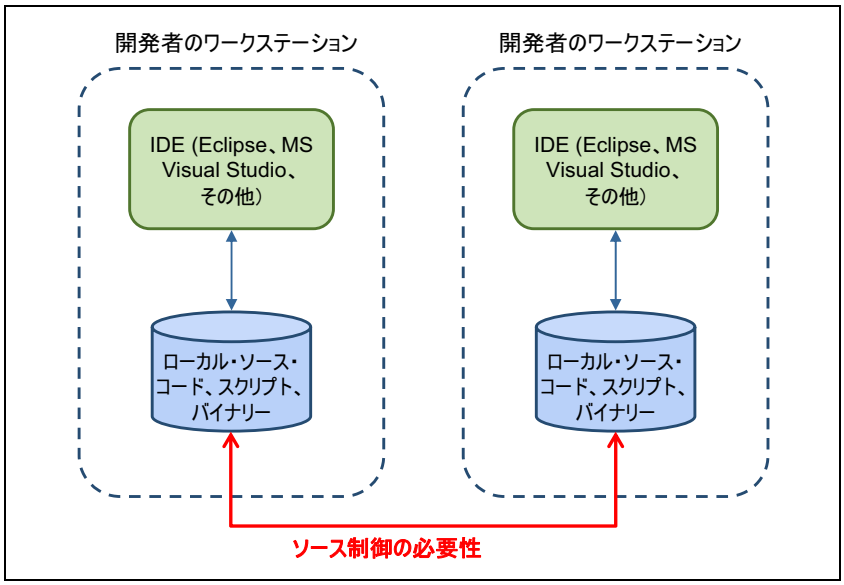


図 4-3　開発者が複数の場合のソース・コード管理の必要性

IDE ベースの開発環境でチームワークを調整するための解決策は、図 4-4 (69 ページ) に示すように、ClearCase または CVS などのソース・コード管理ツールの統合です。現在利用可能なほぼすべてのエンタープライズ・クラスの IDE が、このタイプの統合をサポートしています。

重要な側面は、ソース・コード管理ツールがすべてのソース・コードとバージョン管理の事実上のリポジトリーになることです。IDE の特性は、ソース・コード・ファイルとライブラリーのローカルでの可用性です。IDE は、引き続き、大部分において開発者が 1 名の環境と同じように機能します。編集、コンパイル、およびビルドのプロセスは同じままです。

開発者は、管理オーバーヘッドを発生させることなく、必要に応じていくつでもワークスペースを作成できます。十分なローカル・ディスク・スペースを使用でき、ソース・ファイルがソース管理リポジトリーでチェックインおよびチェックアウトされている限り、開発者は優先される作業をすべて実行できます。

IDE を使用する利点は、ソース・コード管理ツールでのファイルのチェックインとチェックアウトのためのメニューを提供することです。

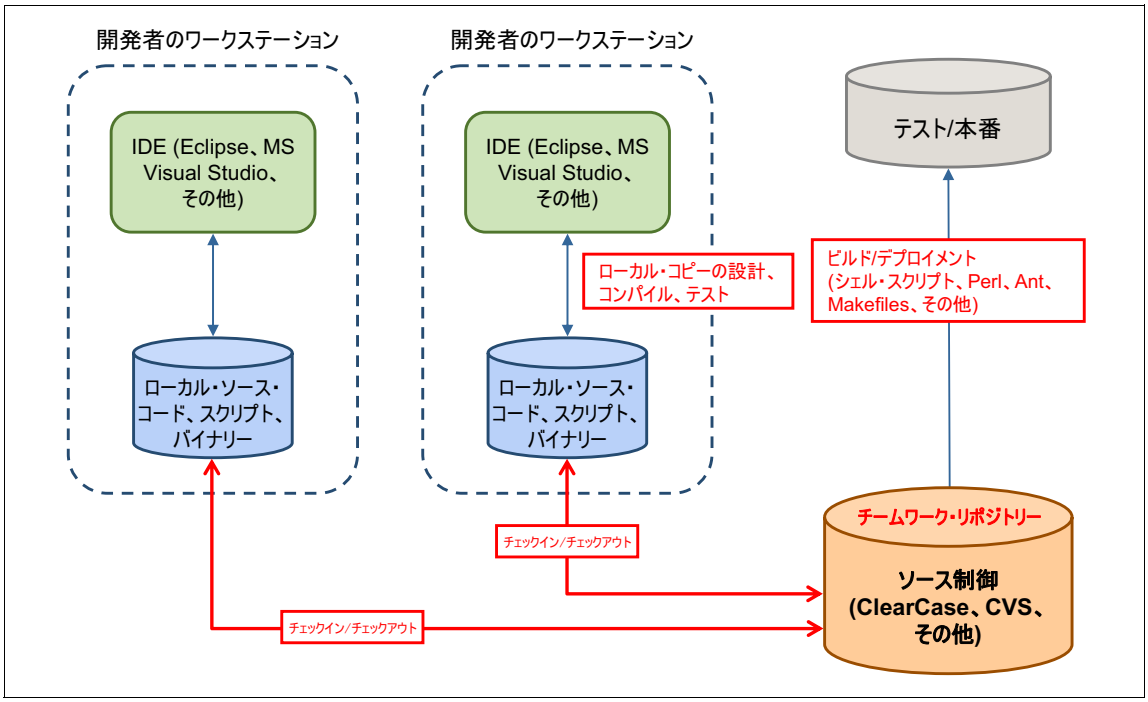


図 4-4　チームワーク・リポジトリーとしてのソース管理

開発者が複数の環境では、*リリース管理者* が登場します。リリース管理者は、リポジトリーから特定のバージョンを引き出して特定のメンテナンス・リリース、つまり 1 つの環境から別の環境にプロモートされる資産のパッケージを作成する担当者です。

通常、特定の基準に従ってバージョンにラベルが付けられます。これにより、リリース管理者の業務は大幅に容易になります。これらのソース・ファイル・バージョンが *ビルド環境* に入れられた後、実際のビルドとテストのプロセスは通常、個々のワークステーションで実行される処理から独立して続行されます。

4.4 Information Server におけるソース制御統合とデプロイメント

InfoSphere DataStage および QualityStage 環境では、これらのツールが提供する管理対象インフラストラクチャーに関連する顕著な相違点がいくつかあります。これらの相違点は、図 4-5 に示すように、開発、ビルド、およびテストのプロセスで反映されています。

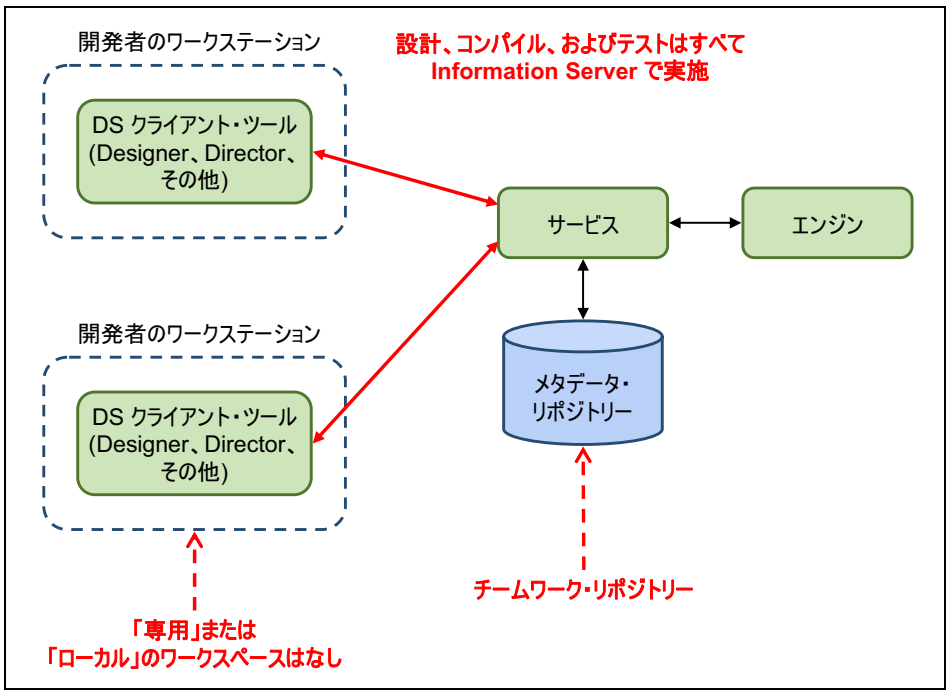


図 4-5　Information Server での開発

DataStage では、すべての処理は、Information Server の各層で直接的に行われ、最終的には同じ Information Server リポジトリーで行われます。各開発者のワークステーションに専用の *ローカル・ワークスペース* はありません。DataStage の設計成果物は、その Information Server リポジトリー (DataStage プロジェクト内) に保管され、すべてのコンパイルとテストは常に Information Server 内部で実行され、一部またはすべての層 (サービス、Information Server リポジトリー、およびエンジン) が関与します。

このアプローチにより、組織は特定の利点を得られます。開発者とビジネス・ユーザーの両方でのコラボレーションが促進され、共通のバックアップと保守の管理プロセスが整備されるためです。

DataStage プロジェクトの作成は、DataStage 管理者が関与する必要がある作業です。この作業では、少なくとも、予期されるプロジェクト・サイズに対応した十分なディスク・スペースが存在することと、許可、所有権、およびグループ・メンバーシップが正しく設定されていることなどを確認する必要があります。

プロジェクトは、通常はビジネス・イニシアチブまたは部門の開発チームと相互に関連する協調的な共有環境です。開発者またはユーザーごとに個別の DataStage プロジェクトを 1 つ作成するのは稀です。

複数の開発者という性質に容易に対応して、各開発者に個々の作業域を提供するために、DataStage での開発の一般的なアプローチには以下のような特徴があります。

- 各開発者の作業を別々の DataStage プロジェクト・フォルダーに分類します。

- サフィックスを付加することにより、同じ*概念的なジョブデザイン*の別々のバージョンを維持します。

作業はフォルダーの点でも構造化され、メジャー・リリースと*増分フィックス*を区別しています。このアプローチは従来からあるものであり、後続のセクションで説明するように標準かつ推奨の方法として考えられます。

図 4-6 (71 ページ) に示すように、ソース・コード管理ツールの統合により、さまざまな資産のバージョンの管理と、チェックインおよびチェックアウトのプロセスが簡素化されます。開発者は、サフィックスに全面的に頼るのではなく、さまざまなバージョンのチェックインとチェックアウトを簡単に実行して、1 つのバージョンを DataStage で維持することができます。

ただし、同じジョブのさまざまなバージョンを同時に処理するには、どの時点でも DataStage プロジェクト内で区別するために、引き続き別々のサフィックスが付けられている必要があります。

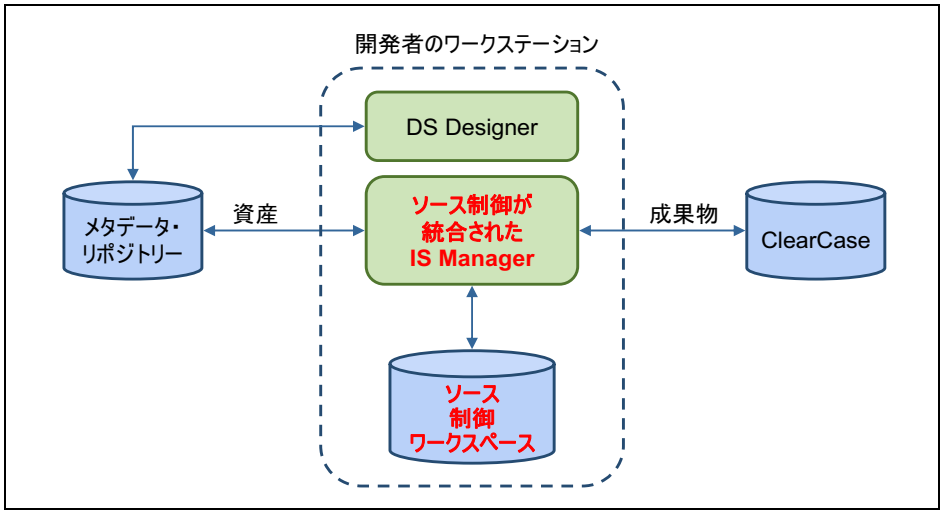


図 4-6　ソース制御が統合された IS Manager

Information Server とのソース・コード管理ツールの統合は、具体的には Information Server Manager (IS Manager) で行われます。

IS Manager は、Eclipse Team Plug-in (IBM Rational Team Concert または ClearCase Remote Client など) を通じて、このタイプの統合をサポートします。Eclipse Team Plug-in は、ソース・コード管理ツールと別のソフトウェア・ベンダー・ツールの間で資産をやりとりするためのワークスペースを念頭に置いて構築されています。統合アプローチは、ソース・コード管理ベンダーが構築した機能を使用する標準モデルに従います。

開発作業は、引き続き DataStage Designer を通じて Information Server リポジトリーに対して (この説明では省略されていますが、サービス層とエンジン層に対しても) 実行されます。

DataStage Designer の中には、チェックインとチェックアウトのプロセスを区別して制御するために使用できるメニューやオプションはありません。すべてのチェックインとチェックアウトは、IS Manager で実行する必要があります。

IS Manager は、デプロイメント・ツールとソース・コード管理の両方の目的で使用されます。

デプロイメントの側面では、ソース・コード管理ツールの存在から比較的独立している IS Manager の機能を利用します。すべてのデプロイメント・パッケージは、常に Information Server リポジトリーから作成されます。ソース・コード管理リポジトリーから直接作成されることはありません。

特定のデプロイメント・パッケージに確実に正しい資産が組み込まれるように、DataStage の開発者は、デプロイメント・パッケージのビルド時に、それらの資産が Information Server リポジトリー内に存在することを確認する必要があります。

図 4-7 に、さまざまな開発者が Information Server リポジトリーとソース・コード管理ツールの両方と対話する様子を示します (このセクションと前のセクションでは、簡潔にするためにサービス層とエンジン層を省略しています)。

Information Server リポジトリーは、ソース制御ツールから完全に独立したままになります。バージョン・ツリー、ラベル、タグなどに関する情報全体が完全にソース・コード管理リポジトリー内に格納されます。

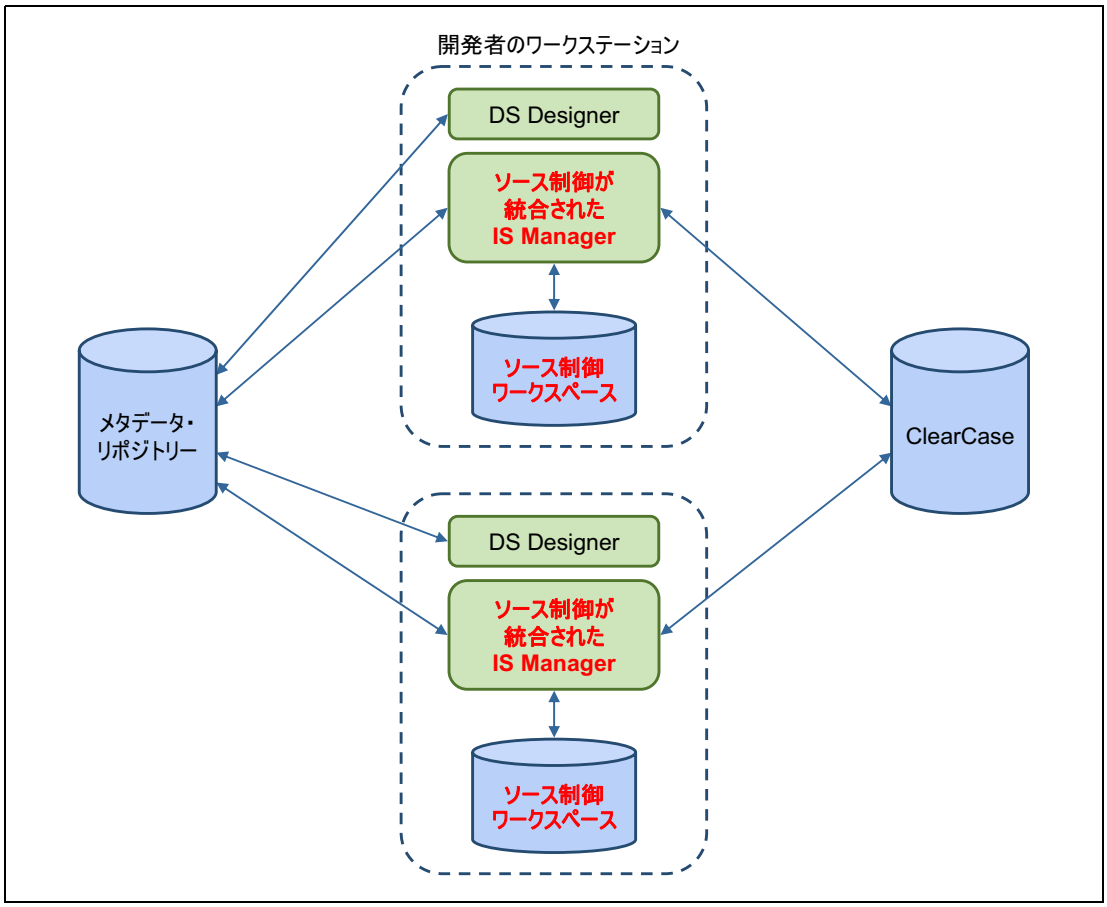


図 4-7　複数の開発者とソース制御統合

4.5 デプロイメントのシナリオと DataStage プロジェクト構造

4.4『Information Server におけるソース制御統合とデプロイメント』(70 ページ) の説明では、DataStage 内で作業を構造化する方法に関する提案の背景を明確にしました。DataStage の開発者は、従来型の IDE とは異なり、完全に専用の開発ワークスペースを持ちません。

すべての設計資産、テスト、実行などは常にサーバー上で直接的に行われます。

そのため、必然的な結果として、同じアプリケーションで同時に作業している多数の開発者に対応するために DataStage プロジェクトを特定の方法で構造化する必要があります。

幸い、旧バージョンの DataStage で採用されたのと同じ優れた手法が引き続き有効であり、ソース管理ツールの統合と完全に互換性があります。大まかに DataStage プロジェクト・フォルダーの作業、リリース、および増分フィックスの編成で構成される同じ規則が引き続き適用されます。

4.5.1 方法

本書では、次の 2 つの主要なデプロイメント・シナリオが究極的に DataStage プロジェクト・フォルダーの構造を決定づけます。

- 完全なリリース候補 (RC)
  - リリース候補ごとに専用の DataStage プロジェクトがあります。
  - 特定のリリース候補を表す DataStage プロジェクトの関連するコンテンツ全体は、特別なフォルダー (release など) に保管されます。
  - release フォルダーのコンテンツ全体が RC デプロイメント・パッケージに組み込まれます。
  - プロジェクトには、リリースに組み込むことがサポートされていないジョブの作業バージョンなど、他のオブジェクトを入れることができます。それらは release フォルダーの外部で保管されます。
  - 各 RC プロジェクトには 1 つの release フォルダーがあります。
  - 開発者は同時に複数の RC バージョンで作業できます。
- 増分フィックス
  - リリース候補のコンテンツの小さなサブセットのみがデプロイメント・パッケージにアセンブルされることが想定されます。
  - 増分フィックスのコンテンツは、特別なフォルダー (IncrementalFixN など) に保管されます。
  - IncrementalFixN フォルダーのコンテンツ全体が特定のフィックス用のデプロイメント・パッケージに組み込まれます。
  - 他のフォルダーに保管されている他のオブジェクトを増分デプロイメント・パッケージに組み込んではなりません。
  - RC プロジェクト内に複数の IncrementalFixN フォルダーが存在してもかまいません。

どちらのシナリオでも、実際の DataStage の資産 (ジョブやシーケンスなど) は、Information Server リポジトリーから直接抽出されます。開発者、リリース管理者、または両方が、コンパイル、テスト、およびリリースの認定を行う正確なバージョンを Information Server リポジトリーにチェックアウトする責任を負います。

デプロイメント・パッケージにアセンブルされるすべての内容が、正常な実行が検証された完全に機能するバージョンであることが前提となります。

新規 RC の作成はメジャー・イベントです。つまり、重大なビジネス上の影響を与えるか、重要な補足コードのリリースをサポートしており、開発、リリース管理、あるいは両方による評価と承認を必要とすることを意味します。例として、EDW_version1、EDW_RC1、SAP_FEED_RC2、SAP_FEED_Phase2 が挙げられます。

新規 RC の作成時には以下の点に注意してください。

- RC がリリースされる時点まで、何回でも更新できます。オブジェクトは複数のバージョンを経由します。
- リリースの後に変更がある場合は、新規リリース候補バージョン (RCn) プロジェクトを作成するか、増分フィックスをリリースすることになります。増分フィックスは、単に、コードのいくつかの部分の改良または軽微な機能拡張です。

*リリース候補プロジェクト*の中では、開発中のオブジェクトとリリースされるオブジェクトを分離するために別々のフォルダーを使用することが推奨される手法です。

完全なリリース候補に組み込むものと組み込まないものに関する情報を維持するためにソース制御ツールに依存するのは、優れた手法ではありません。RC の一部として組み込むものと組み込まないものの問題に対応するには、オブジェクトをフォルダーで編成します。この方法では、全員が、オブジェクトがまだ開発段階にあることや、リリース候補にも組み込んではならないことを即時に理解できます。

1 つのフォルダーに release という名前を付けます。このフォルダーに格納されているすべてのものは、*最新*と見なされ、完全な*リリース候補*の一部になります。

図 4-8 は、リリース候補ごとに 1 つずつ、複数の DataStage プロジェクトを示すダイアグラムです。各プロジェクトの中に、1 つの release フォルダーと 1 つ以上の IncrementalFix フォルダーがあります。開発中の資産はさらに個別のフォルダーに分離できます。

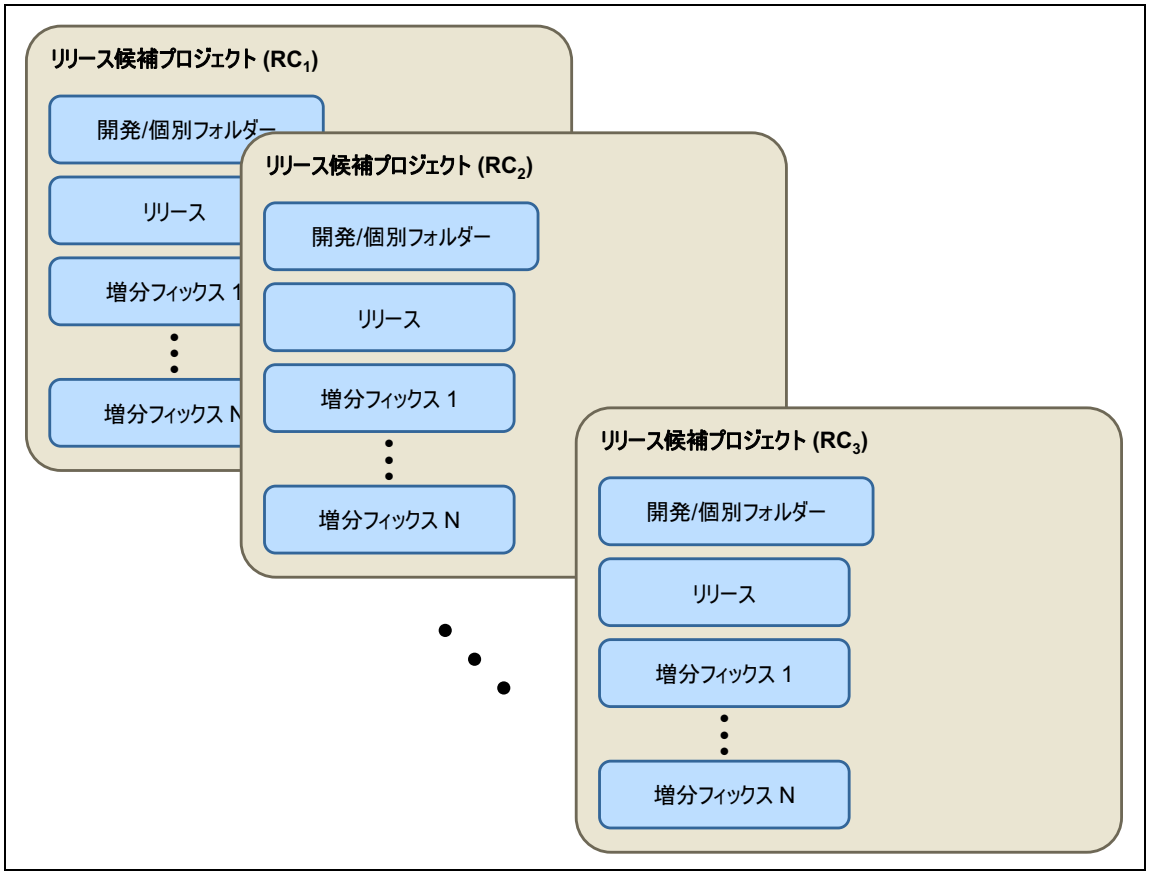


図 4-8　リリース候補と増分フィックス

図 4-9 は、リリース候補プロジェクトとデプロイメントおよびソース管理の側面との間の関係を示しています。また、1 つのフォルダーから別のフォルダーへの資産の移動も示しています。

特定のフォルダー内の資産の存在は、資産の状態を明確に特徴付けています。この分かりやすい指標は、開発者とリリース管理者の両方のためのものです。ソース制御ラベルまたはタグにのみ依存するのではなく、開発者と管理者のコミュニティー全体で情報が可視化されます。

前述のとおり、これらの手法は、旧バージョンの DataStage で採用されたのと同じ手法であり、バージョン 8.5、8.7、およびそれ以降でも適用されます。

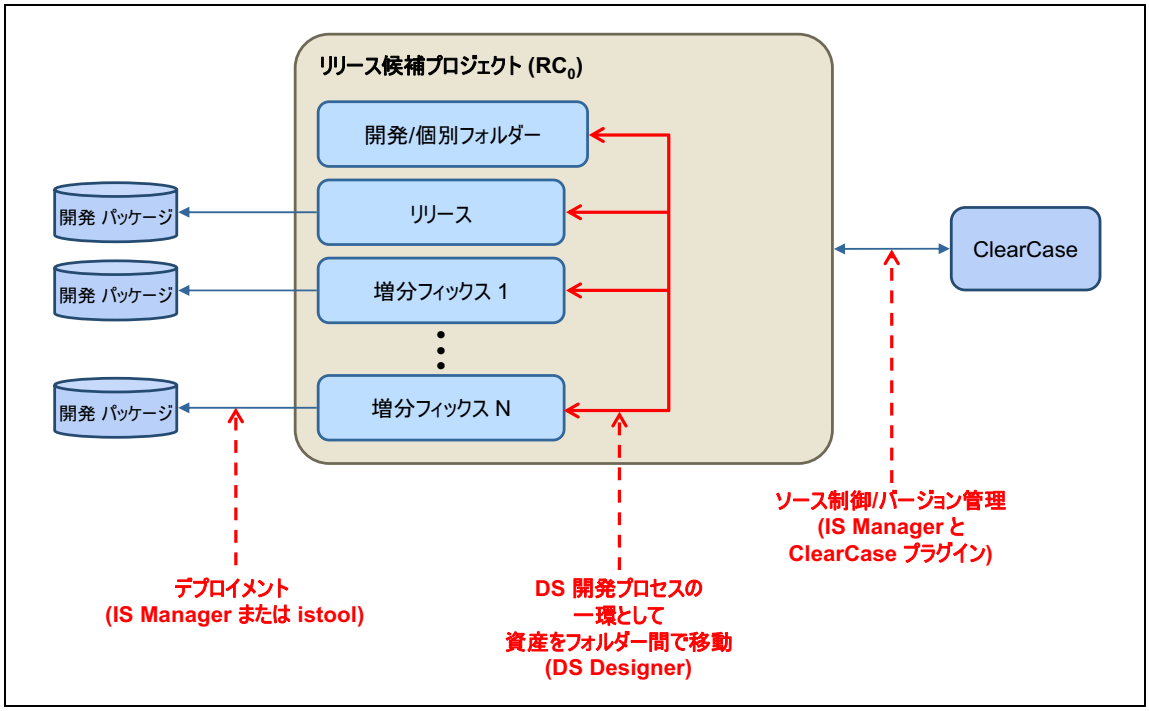


図 4-9　デプロイメントとソース制御のコンテキストにおけるリリース候補

4.5.2 方法の概要

下記のリストに、このセクションでのいくつかの前提と結論をまとめます。

- いずれの新規リリース候補でも、別個の DataStage プロジェクトを作成する必要があります。新規リリース候補の RCn プロジェクトの開始点は、RCn-1 のコピーとなる傾向があります。
- デプロイメント・パッケージは常に全体として ClearCase にチェックインされます。
- デプロイメント・パッケージは、ClearCase ではなく、常に IS Manager または istool を介して DataStage から作成されます。
  - 必ず、オブジェクトがコンパイルされていて、RC 全体が正常に機能している必要があります。
  - これらは確実に同期されるように一緒に抽出されます。
- 文書化の目的で便利な手段として ClearCase ラベルを設定できます。これにより、どのオブジェクト・バージョンが RC に組み込まれているか分かりやすくなります。ただし、デプロイメント・パッケージに入れられる実際のソースは、正常な実行とテストが確認された後で、DataStage プロジェクトから直接入れられます。これらのオブジェクトは、リリース候補自体の DataStage プロジェクト内の release フォルダーから削除される必要があります (同じことが増分フィックスにも当てはまります)。
- 新規 RC の作成は、開発およびリリース管理による評価と承認を必要とするメジャー・イベントです。
  - RC がリリースされる時点まで、何回でも更新できます。オブジェクトは複数のバージョンを経由します。
  - リリースの後に変更がある場合は、新規 RCn プロジェクトを作成するか、増分フィックスをリリースすることになります。

4.6 IS Manager および ClearCase によるソース管理およびバージョン管理

開発者は、ジョブデザインおよびその他のアイテムの作成、更新、および削除のために DataStage Designer を使用する必要があります。ただし、デプロイメント・パッケージの作成とバージョン管理の作業を行う場合、これらの作業のためのツールは Information Server Manager になります。

図 4-10 は、開発者がジョブデザインを作成して ClearCase にチェックインするために必要な下記のステップを示しています。

1. DataStage Designer を使用してジョブを作成および更新します。開発者が ClearCase に実行依頼できる状態であると判断した後、次の 2 つのステップでは IS Manager が使用されます。
2. ジョブデザインをローカル・ソース管理ワークスペースに移動します。
3. ジョブを ClearCase にチェックインします。

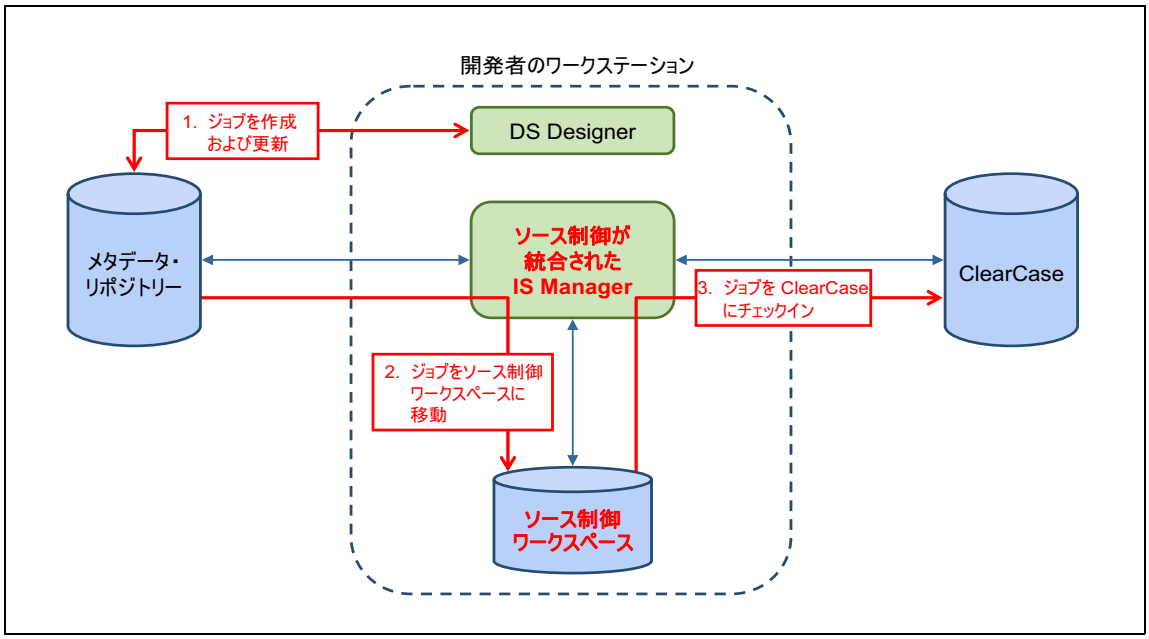


図 4-10　ジョブの作成と ClearCase へのチェックイン

図 4-11 は、図 4-10 (77 ページ) を拡張したものであり、以下のステップにより、ジョブを ClearCase からチェックアウトして、DataStage Designer で更新し、ジョブを ClearCase に戻してチェックインするプロセスを示しています。

1. IS Manager 内部から ClearCase Remote Client Plug-in を使用して、ジョブが ClearCase からチェックアウトされます (その結果、このジョブのコピーが、開発者のワークステーションのローカル側にあるソース管理ワークスペースに保存されます)。
2. 開発者は、IS Manager を使用して、ソース管理ワークスペースに保管されているバージョンによって Information Server リポジトリーを更新します。
3. DataStage Designer により、ジョブの更新、コンパイル、およびテストが行われます。すべての更新は、Information Server リポジトリーで直接的に行われます。開発者は、ジョブを ClearCase に戻してチェックインすることが妥当になる時点まで、必要な時間にわたってジョブをチェックアウトしたままにすることができます。
4. 更新のステップが終了した後、開発者は、このジョブの新規バージョンのコピーを Information Server リポジトリーからソース管理ワークスペースに移動します。ステップ 1 および 2 と同様、この作業も IS Manager で行われます。

5. 引き続き IS Manager で、開発者は Remote Client Plug-in を使用してジョブを ClearCase に戻してチェックインします。

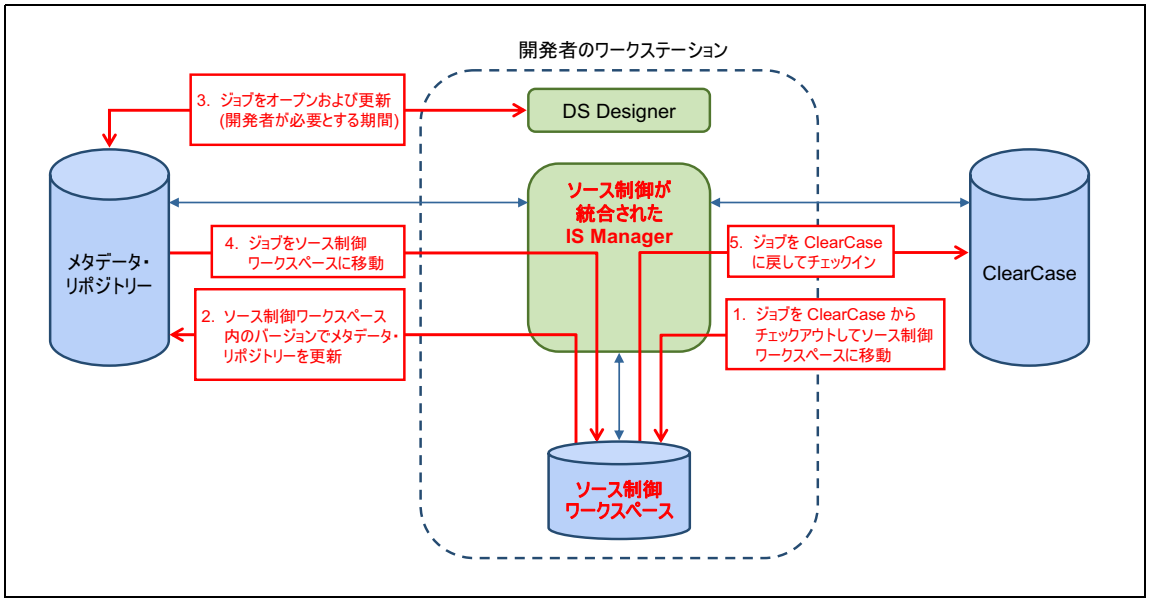


図 4-11　チェックインとチェックアウトのサイクル

4.7 IS Manager によるデプロイメント

図 4-12 は、IS Manager パッケージ・エディターの「*Send*」オプションを使用してパッケージをプロモートするプロセスを示しています。

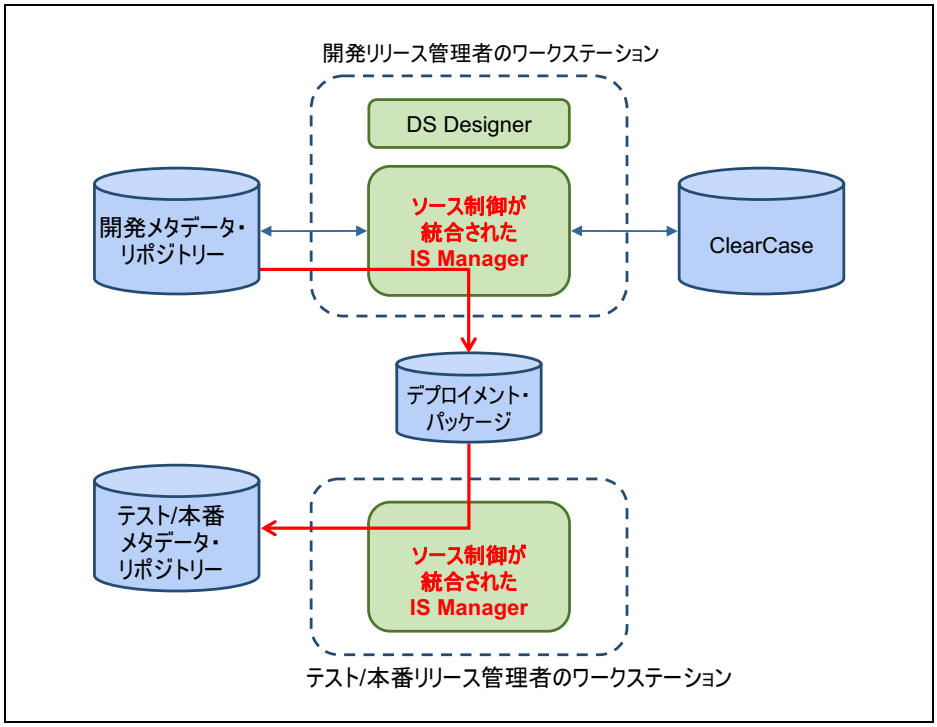


図 4-12　「Send」オプションを使用するデプロイメント・パッケージのプロモート

図 4-13 図 4-13『ClearCase のチェックイン / チェックアウトを使用するデプロイメント・パッケージのプロモート』（79 ページ）は、IS Manager ClearCase の

パースペクティブを介してパッケージをプロモートするプロセスを示しています。

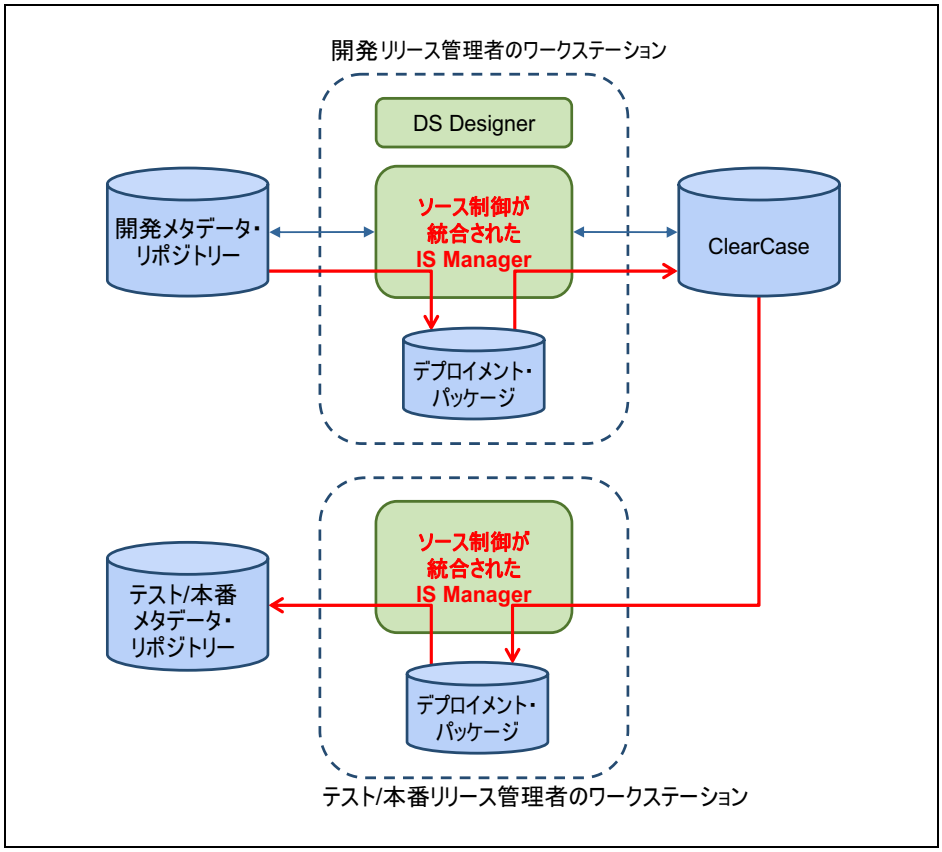


図 4-13　ClearCase のチェックイン / チェックアウトを使用するデプロイメント・パッケージのプロモート

図 4-14 (80 ページ) は、各リリース候補バージョン (RCn) のために作成されたデプロイメント・パッケージを ClearCase にチェックインするプロセスを示しています。この図は、複数の RC バージョン (各 RCn バージョンは別々の DataStage プロジェクト) を作成するプロセスに関連しています。

デプロイメント・パッケージを ClearCase にチェックインする利点は、リリース管理者とテスト管理者が、後続のテスト段階や本番で進行状況とリリースを文書化するためにコメントまたはラベルを適用できることです。

このアプローチにより、資産のバージョンを追跡できるだけでなく、バックアップ / リストアのどの時点にも戻ることができる堅固な環境の再作成メカニズムを実現することもできます。

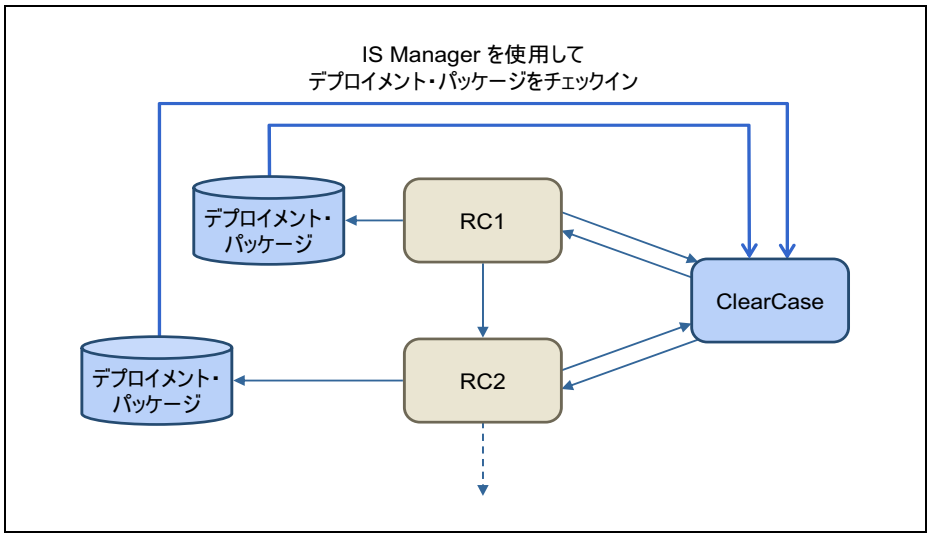


図 4-14　デプロイメント・パッケージのチェックイン (ClearCase のみ)

この図には、テスト環境へのデプロイメント・パッケージのロードは示されていません。この図は、デプロイメント・パッケージを ClearCase にチェックインするためのツールとしての IS Manager の使用法のみを示しています。

4.8 リリース候補が複数の場合のワークフロー

このセクションでは、4.5『デプロイメントのシナリオと DataStage プロジェクト構造』（72 ページ）にリストされている前提に従って、DataStage アプリケーションで複数のリリース候補を扱うシナリオについて説明します。

ライフサイクル管理プロセスは、DataStage アプリケーションの開発環境、テスト環境、および本番環境を使用して実装できます。

標準的な手法では、ライフサイクルの各段階に別々の物理環境を用意します。開発システム、テスト・システム、および本番システムには別々の物理環境があります。

また、開発システムとテスト・システムが単一の物理環境を共有して、本番システムが別の物理環境に置かれる混合システムも実装できます。

推奨としては、開発システム、テスト・システム、および本番システムを共通の物理環境で共有*しない*形になります。また、変更は開発環境*のみ*で*実施*し、後続の段階や環境に伝搬します。

下記のリストに示すようにさらに多くの段階や環境が伴うシステム・ライフサイクルも、比較的一般的です。

- 開発と保守
- 統合テスト
- 品質保証
- ユーザー受け入れテスト
- 本番

この章と後続の章では、例として以下の DataStage 環境を取り上げます。

- 開発 (DEV)
- システム統合テスト (SIT)
- 技術テスト (TT)
- 本番 (PROD)

図 4-15 は、リリース候補が複数の場合のワークフローを示しています。

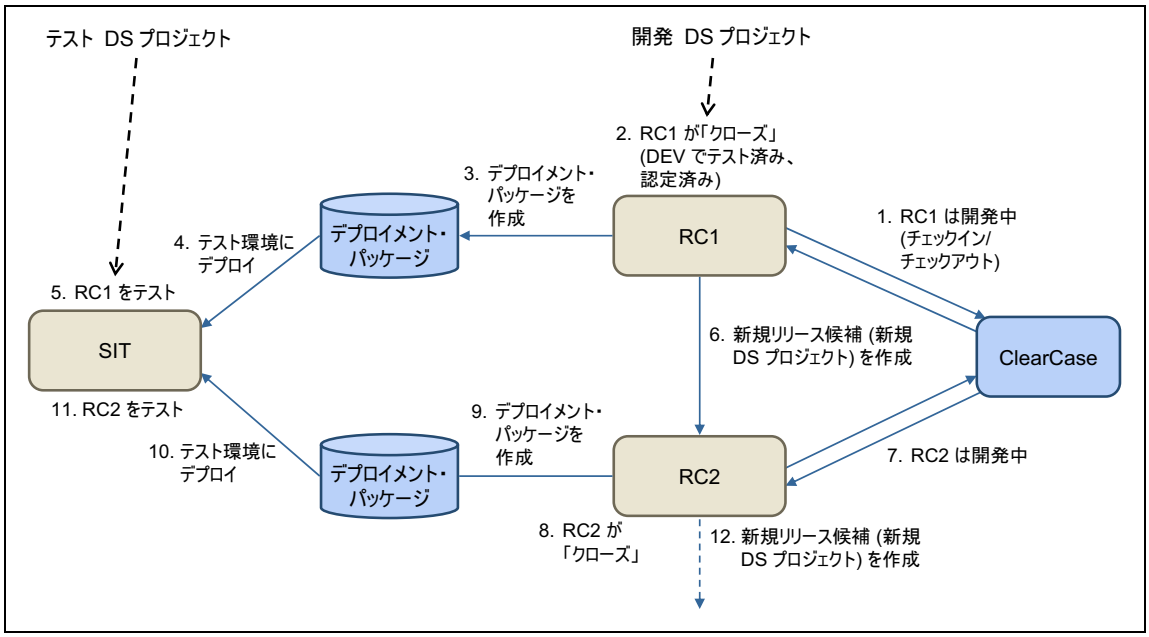


図 4-15　リリース候補が複数の場合のワークフロー

図 4-15 (81 ページ) には、以下のステップが示されています。

1. RC1 は開発中です。開発者は、IS Manager を使用して ClearCase でオブジェクトのチェックインとチェックアウトを行い、必要に応じてオブジェクトを作成および除去し、個々の単体テストを実行します。
2. RC1 は、完全にテストされ、後続の環境へのリリースに認定されます。
3. 標準化された一連のフォルダー (release フォルダーなど) から選択することにより、RC1 用のデプロイメント・パッケージが作成されます。
4. デプロイメント・パッケージがターゲットのテスト環境にロードされます。
5. RC1 は SIT でテストされます。ステップ 4 および 5 が TT に対して繰り返され、すべてが正常な場合は本番に対して繰り返されます。
6. 新しい RC が作成されます (RC2)。このステップは、フィックスや追加機能が必要になった結果として実行される場合があり、RC2 用に新しい DataStage プロジェクトが作成されたことを意味します。
7. RC2 は開発中です。開発者は、IS Manager を使用して ClearCase でオブジェクトのチェックインとチェックアウトを行い、必要に応じてオブジェクトを作成および除去し、個々の単体テストを実行します。
8. RC2 は、完全にテストされ、後続の環境へのリリースに認定されます。
9. 標準化された一連のフォルダーから選択することにより、RC2 用のデプロイメント・パッケージが作成されます。
10. デプロイメント・パッケージがターゲットのテスト環境にロードされます。
11. RC2 は SIT でテストされます。ステップ 4 および 5 が TT に対して繰り返され、すべてが正常な場合は本番に対して繰り返されます。
12. 新しい RC が作成されます。このサイクルが繰り返されます。

4.9 Information Server のデプロイメントおよびバージョン管理ツールのコンテキスト

図 4-16 は、個別のテスト環境と検証環境へのパッケージのデプロイメントと、各 RC の開発で行われるバージョン管理の分離を示しています。

各プロジェクト内のバージョン管理ソリューションとして ClearCase が使用されます。

デプロイメント・パッケージの作成は、IS Manager (または場合によっては istool ユーティリティー) で行われます。パッケージはファイル・システムに保存されてターゲットのテスト環境にロードされますが、この場合も IS Manager が使用されます

4.8『リリース候補が複数の場合のワークフロー』（80 ページ）で説明したシナリオとは異なり、図 4-16 では、デプロイメント・パッケージは ClearCase にチェックインされません。

各リリース候補に別々の DataStage プロジェクトを作成する際、4.5『デプロイメントのシナリオと DataStage プロジェクト構造』（72 ページ）にリストされている前提に従います。新規リリース候補の作成は、新しい DataStage プロジェクトの作成を意味するメジャー・イベントです。

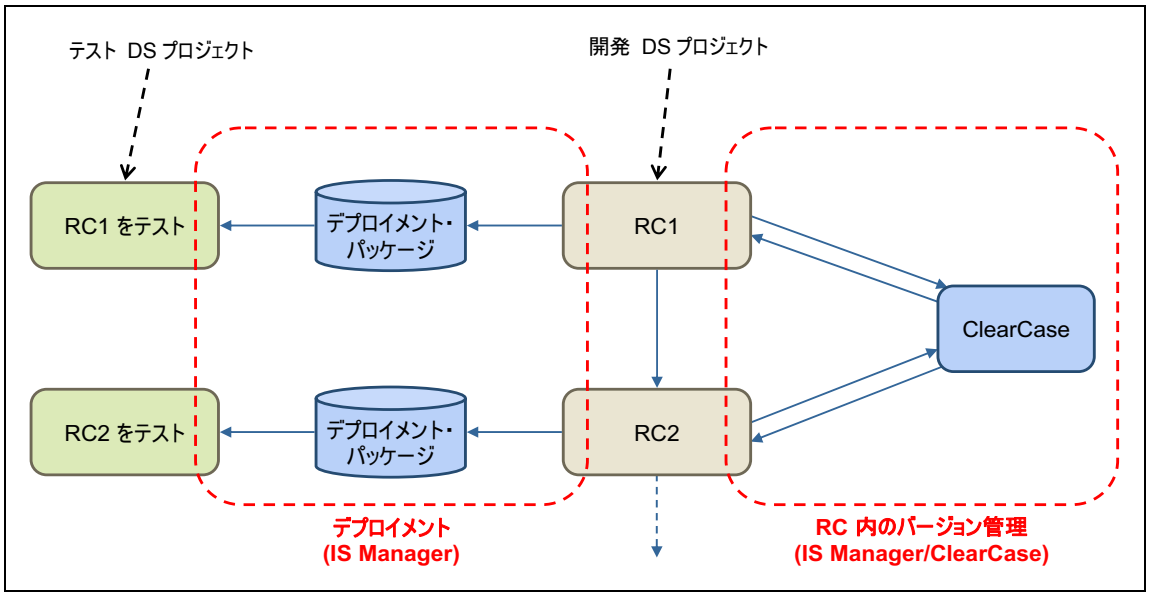


図 4-16　IS のデプロイメントおよびバージョン管理ツールのコンテキスト

4.10 本番へのプロモーションのためのワークフロー

図 4-17 は、開発から本番までのさまざまな環境で正常にプロモートおよびテストされるデプロイメント・パッケージのワークフローの概要を示しています。

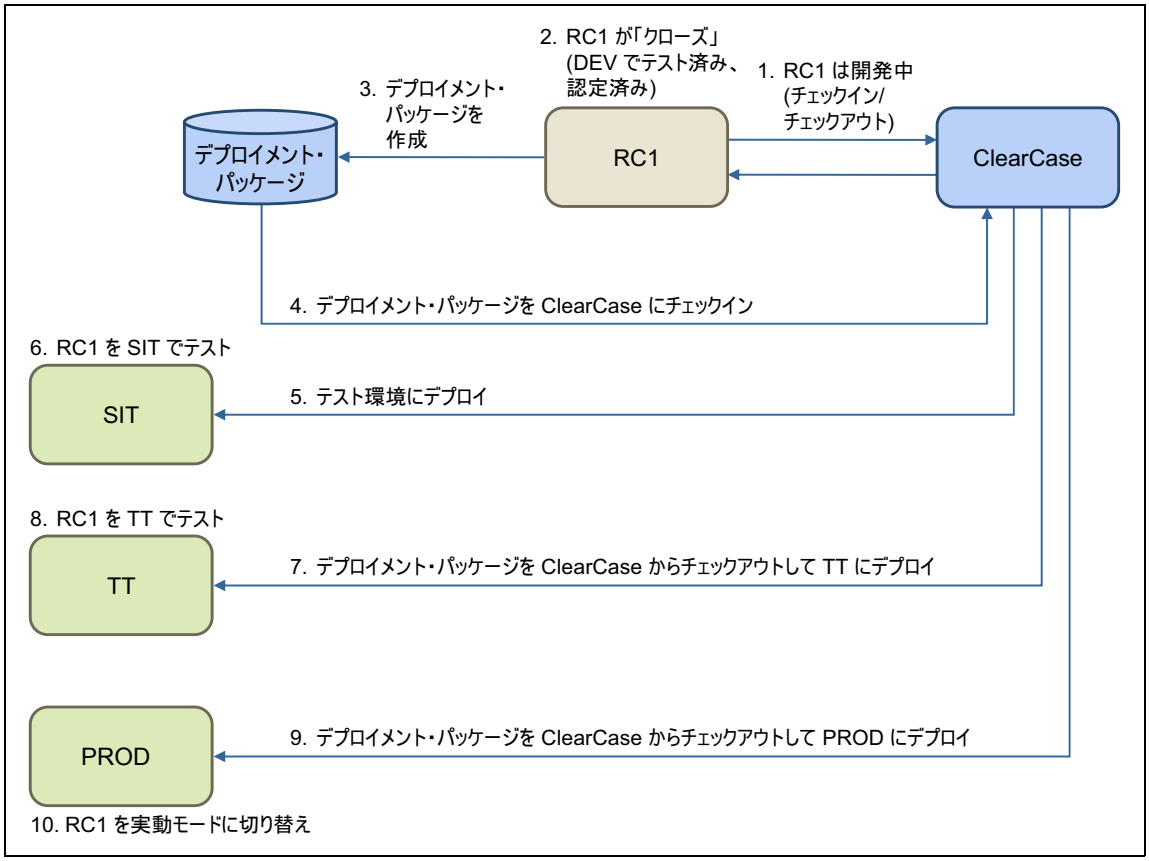


図 4-17　リリース候補を本番に配置するためのワークフロー

IS Manager および ClearCase Eclipse プラグインを使用するこのプロセスの実装については、4.20『デプロイメント・パッケージの作成とビルド』（120 ページ）および 4.22『ClearCase Explorer によるデプロイメント・パッケージのプロモート』（125 ページ）で説明しています。

図 4-17 (83 ページ) には、以下のステップが示されています。

1. RC1 は開発中です。開発者は、IS Manager を使用して ClearCase でオブジェクトのチェックインとチェックアウトを行い、必要に応じてオブジェクトを作成および除去し、個々の単体テストを実行します。
2. RC1 は、完全にテストされ、後続の環境へのリリースに認定されます。
3. RC1 用のデプロイメント・パッケージが作成されます。このステップには DataStage オブジェクト (ジョブ、シーケンス・パラメーター・セットなど) が含まれます。
4. デプロイメント・パッケージは ClearCase にチェックインされます。
5. パッケージはチェックアウトされ、SIT にデプロイされます。
6. RC1 は SIT でテストされます。
7. テストは正常に行われ、ClearCase で SIT 管理者によって注釈が付けられます。RC1 デプロイメント・パッケージは、ClearCase からチェックアウトされ、TT にデプロイされます。
8. RC1 は TT でテストされます。

9. テストは正常に行われ、ClearCase で TT 管理者によって注釈が付けられます。デプロイメント・パッケージは、再び ClearCase からチェックアウトされ、本番にロードされます。
10. RC1 は本番モードに切り替わります。

4.11 ビルド環境

DataStage ジョブのコンパイルは、DataStage クライアント・システムを使用する必要があるプロセスです。コンパイルは主に DataStage Designer 側で行われますが、Designer はコンパイル中に必然的にサービス層、エンジン層、および Information Server リポジトリー層と対話します。そのため、既存の個々の開発者または管理者のワークステーションを使用するか、共有環境でこの目的のための専用マシン (Citrix など) を割り振ることにより、毎晩のバッチ・ビルドを実行する必要があります。

コンパイルは、以下のいずれかのアプローチによってトリガーされます。

- コマンド・ライン・コンパイラー dscc.exe

  dscc.exe はフォルダーを引数として受け入れますが、サブフォルダーは含まれません。すべてのサブフォルダーは明示的に参照される必要があります。

- DataStage Designer からアクセスできるバッチ・コンパイラー

  DataStage Designer のバッチ・コンパイラーは、すべてのサブフォルダーを含むフォルダー全体を選択できるため、さらに便利な手段となります。

したがって、標準的な手法では、毎晩のビルドのために DataStage Designer のバッチ・コンパイラーを採用して、共有されている Citrix 環境のマシンからこのツールを呼び出します。このマシンは、可能な限り、IS インスタンスに近い場所に配置する必要があります。このタイプのアクティビティーには多くのネットワーク・トラフィックが伴うためです。

4.12 IS Manager を使用したドメインへの接続

このセクションでは、Information Server ドメインへの接続を確立するプロセスについて説明します。このセクションの図 (図 4-18 から図 4-21 (87 ページ)) には、以下の情報が示されています。

- Windows デスクトップ上で IS Manager アイコンを見つける方法
- 新規ドメイン定義を追加して、ユーザー資格情報を入力し、接続をテストする方法
- 正常にドメインに接続した後の「Repository」ビューの外観

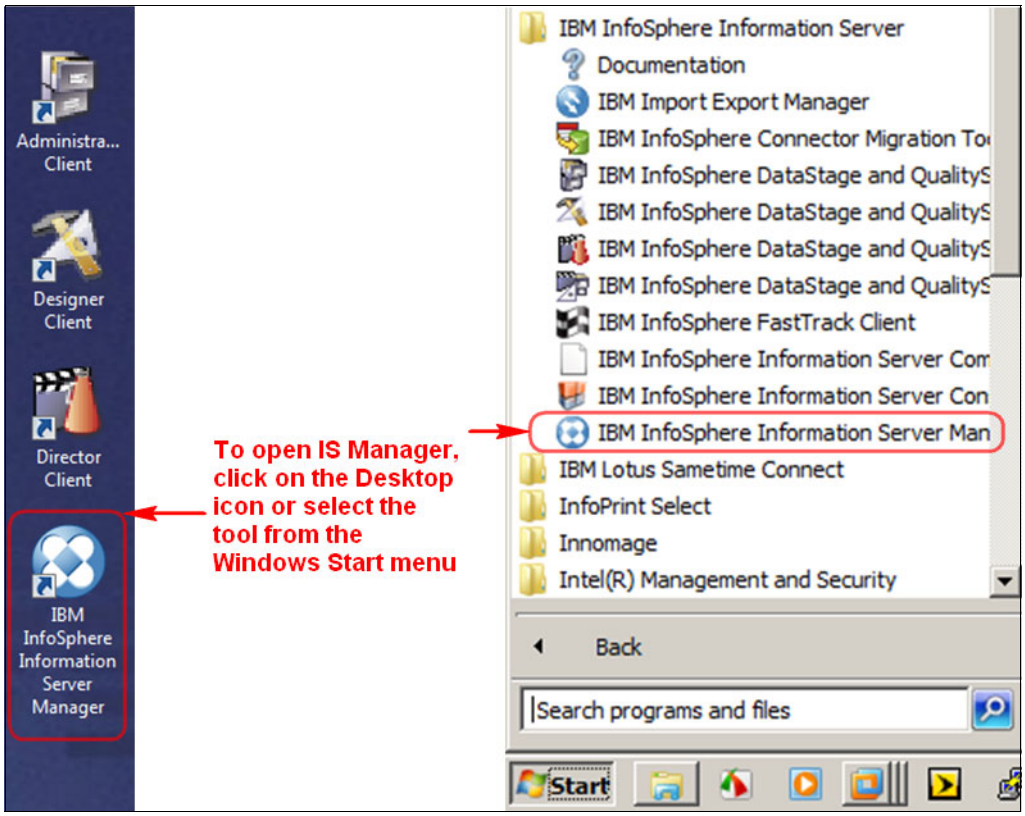


図 4-18　Windows デスクトップでの IS Manager アイコンの位置確認

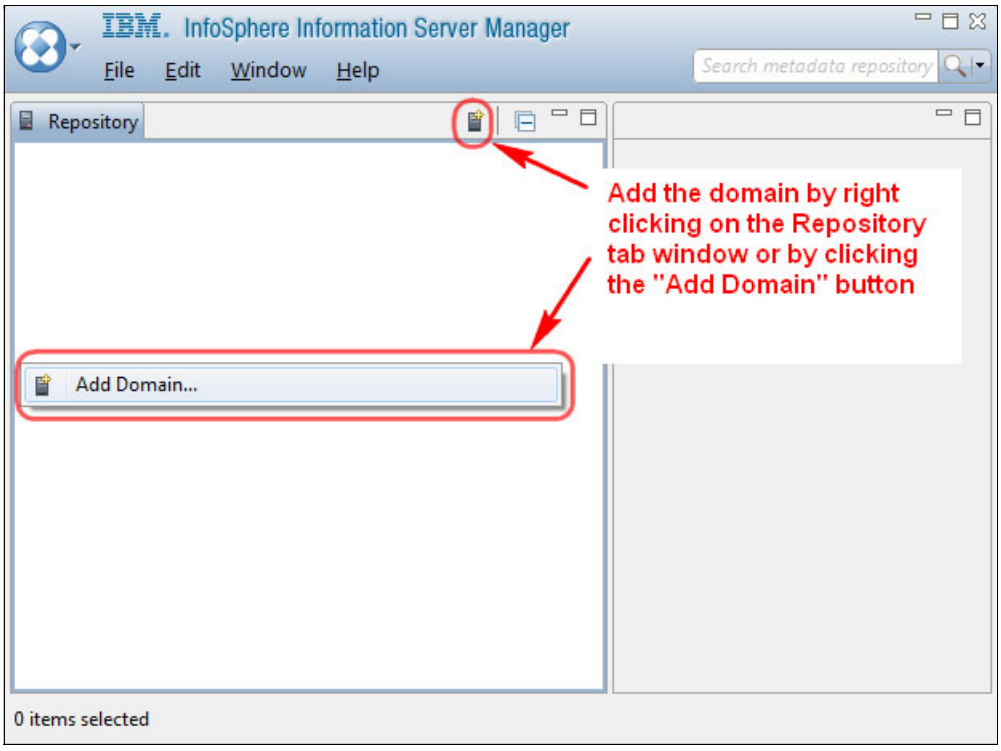


図 4-19　「Add Domain」オプションの選択

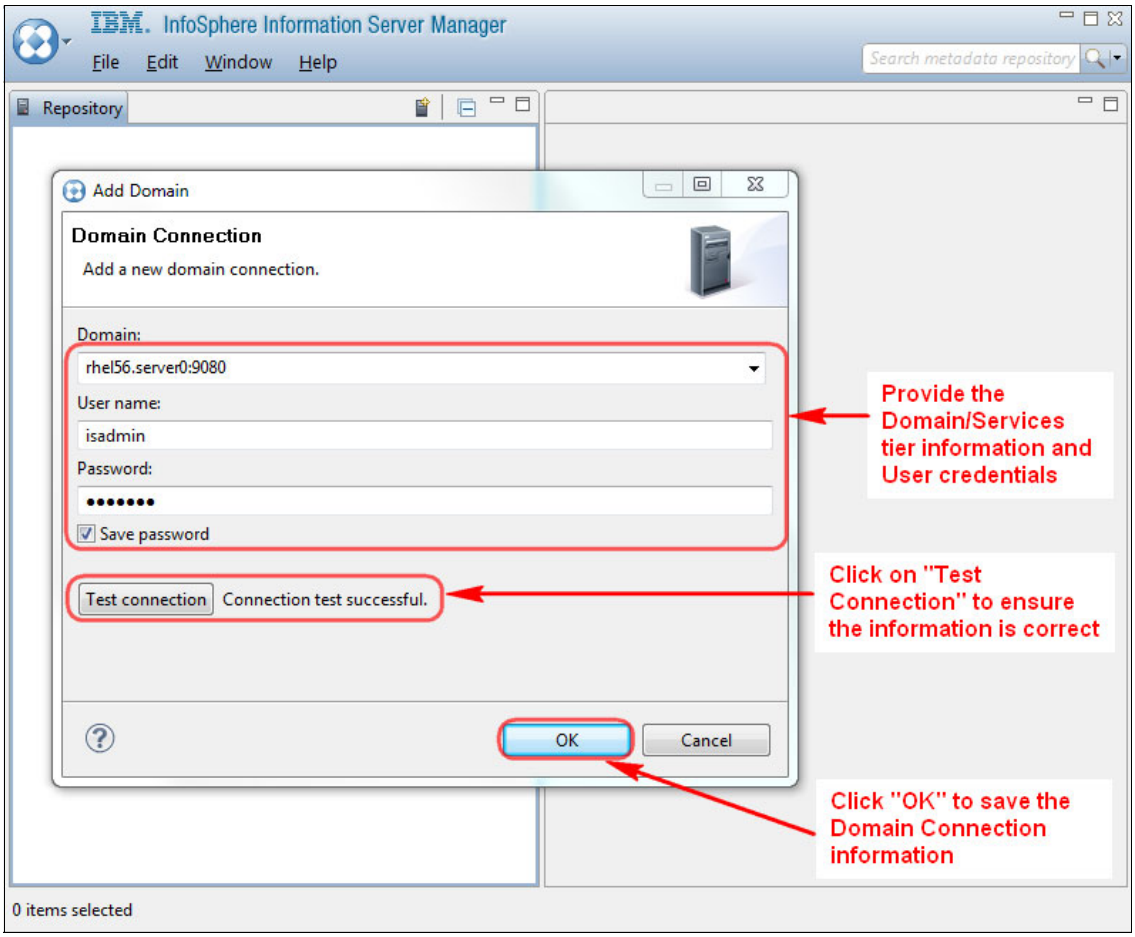


図 4-20　IS ドメインの定義

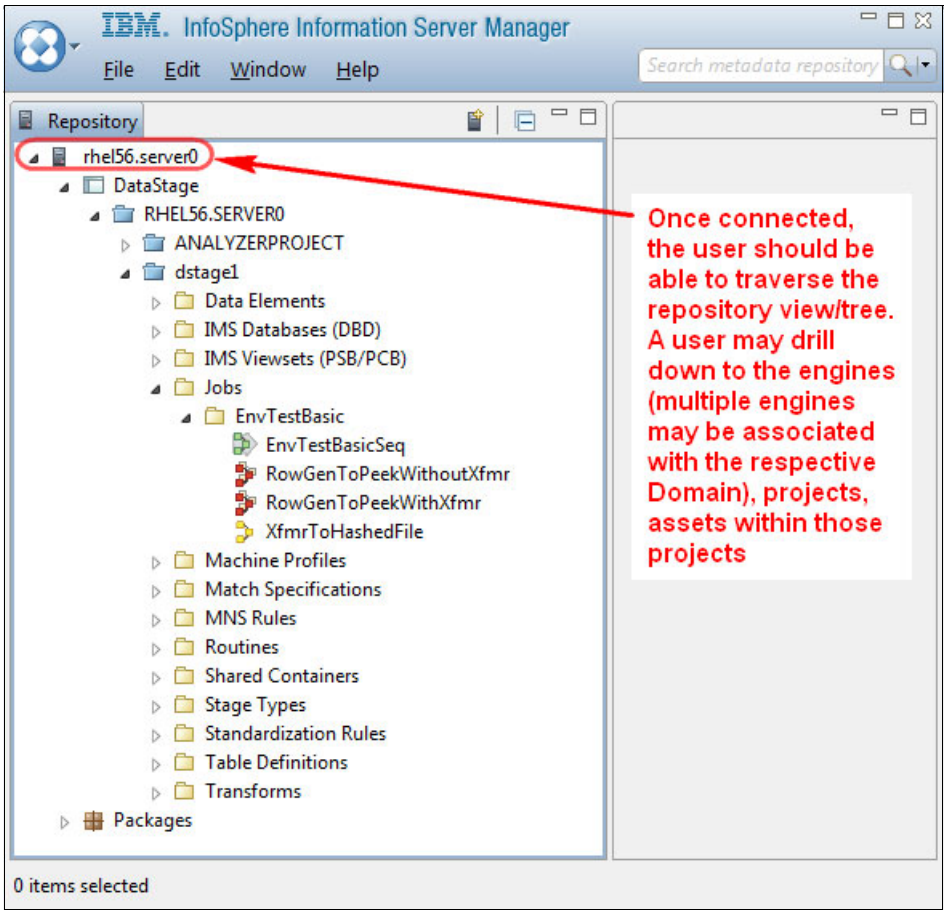


図 4-21　正常に接続した後の Information Server リポジトリーのビュー

4.13 IBM Rational ClearTeam Explorer Eclipse プラグインのインストール

これで IS Manager で IS ドメインが定義されたため、次のステップでは、ソース管理ツールの Eclipse プラグインをインストールします。この場合、ソース管理ツールとして ClearCase を使用しているため、プラグインには以下のいずれかの名前が付けられています。

- IBM Rational ClearTeam Explorer for Eclipse プラグイン
- IBM Rational ClearCase Remote Client for Eclipse プラグイン

IBM Rational ClearTeam Explorer は、Rational ClearCase のソース制御下でリソースにアクセスしてそれらを変更するために使用されます。IBM Rational ClearTeam Explorer は、Eclipse テクノロジーを採用したユーザー・インターフェースであり、開発者によってローカル・エリア・ネットワーク (LAN) または広域ネットワーク (WAN) の両方を介して使用されます。

このセクションの図 (図 4-22 (89 ページ) から図 4-28 (92 ページ)) には、ClearTeam Explorer for Eclipse プラグインのインストール・プロセスを開始するための以下の方法が示されています。

1. 「**Help**」→「**Install New Software**」を選択することによってインストールを開始します。
2. 「**Add**」をクリックすることによってソフトウェア・サイトを追加します。
3. 名前とロケーションを追加することによってサイトの詳細情報を指定します。
4. インストールするフィーチャーを選択します。
5. インストールの詳細を確認して、「**Next**」をクリックします。
6. ライセンスを確認して同意し、「**Next**」をクリックします。
7. 「**Restart Now**」をクリックして、Information Server Manager を再起動します。

名称 : 旧バージョンの ClearCase では、IBM Rational ClearTeam Explorer for Eclipse プラグインは Rational ClearCase Remote Client for Eclipse プラグインと呼ばれることもあります。

このプロセスは、ソース管理の統合機能を利用予定のすべてのユーザーの各クライアント・ワークステーションで実行する必要があります。

その他のソース・コード管理システム用のプラグインをインストールするためのステップも似ています。

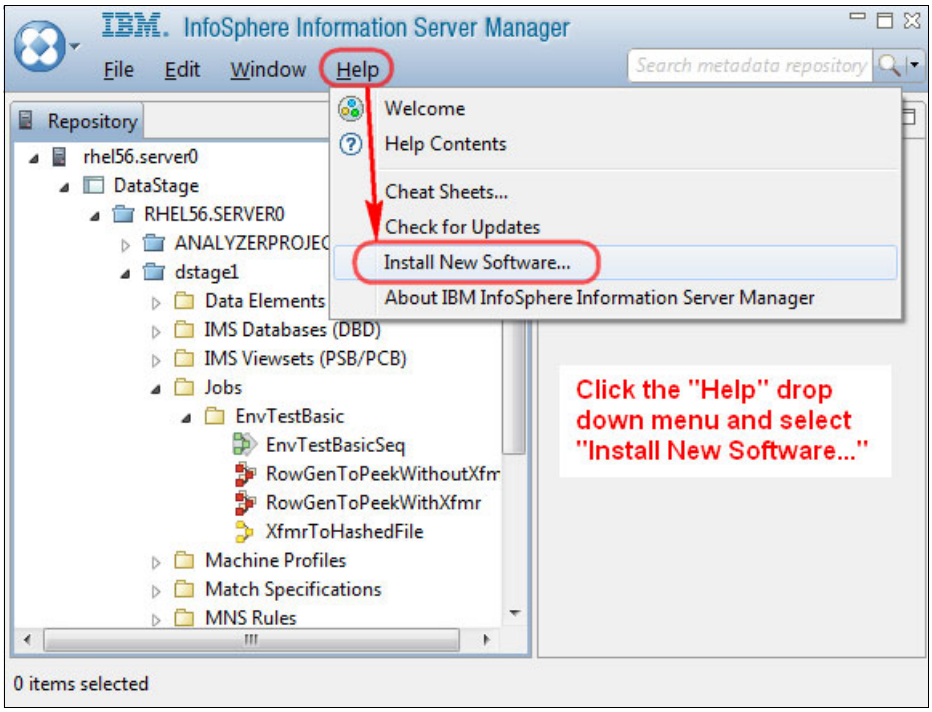


図 4-22　Eclipse プラグインのインストール・プロセスの開始

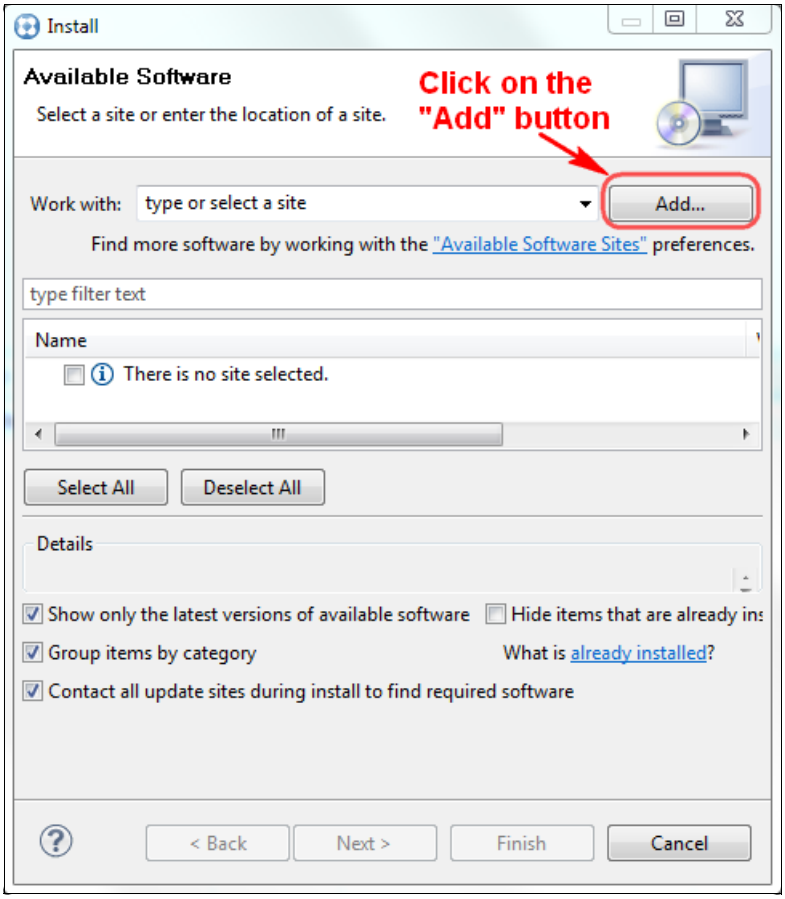


図 4-23　Eclipse プラグインをダウンロードするためのソフトウェア・サイトの追加

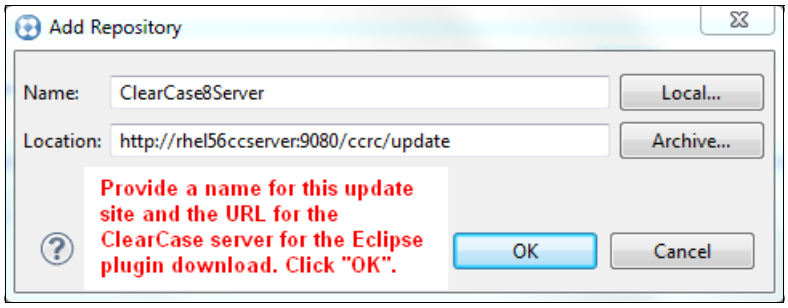


図 4-24　ClearCase サーバー (ソフトウェア・サイト) の詳細情報の指定

図 4-24 (90 ページ) で、「Add Repository」ダイアログの「Name」フィールドに任意の希望の名前を指定できることに注意してください。ただし、「Location」フィールドには以下の形式を使用してください。<HostName> は、ClearCase サーバーのホスト名です。

http://<HostName>:9080/ccrc/update

図 4-24 (90 ページ) に示されている値は、単なる例です。ご使用の環境の実際の ClearCase サーバーに適用されるホスト名を使用してください。

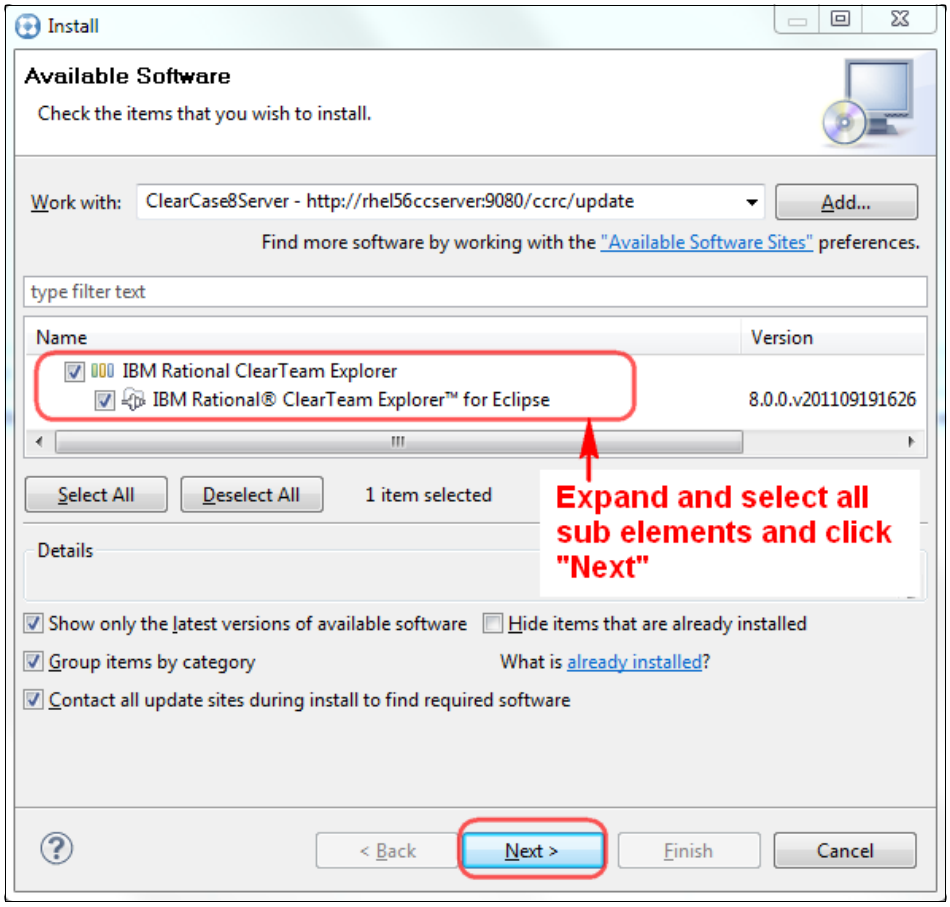


図 4-25　インストールする Eclipse プラグイン・フィーチャーの選択

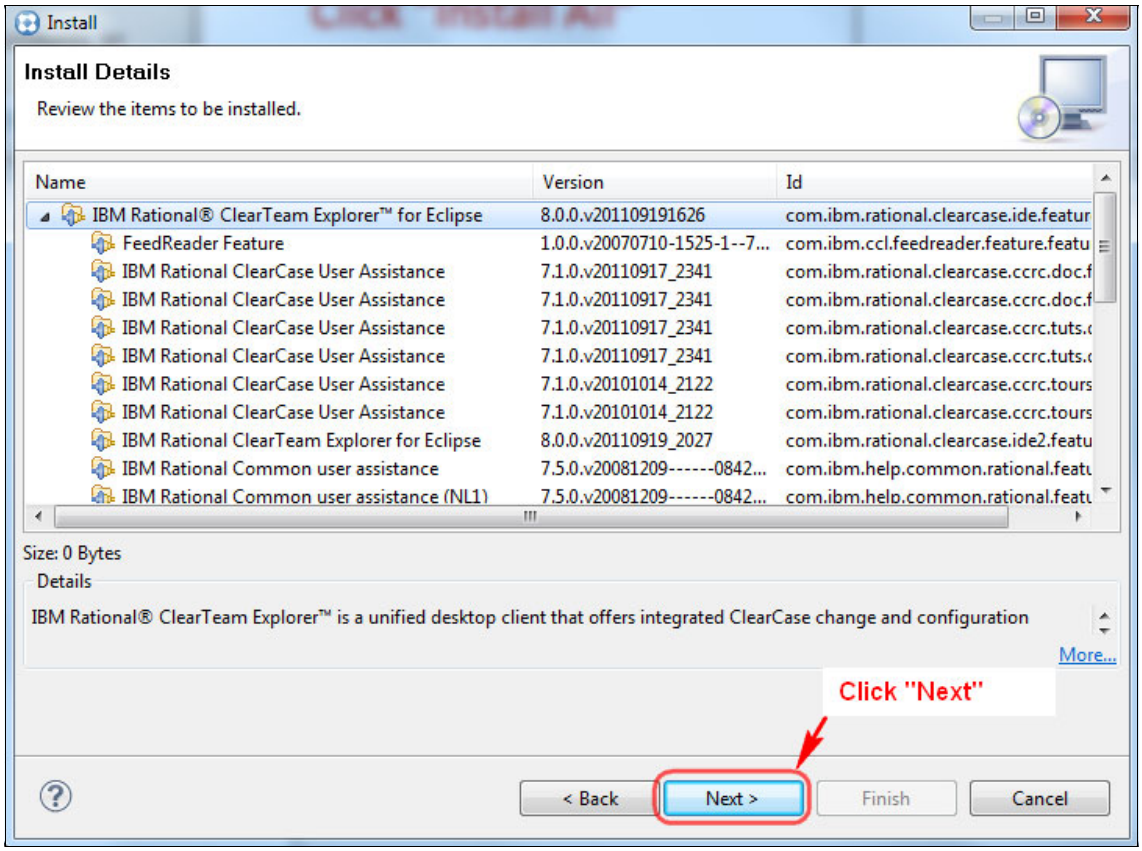


図 4-26　確認のパネル

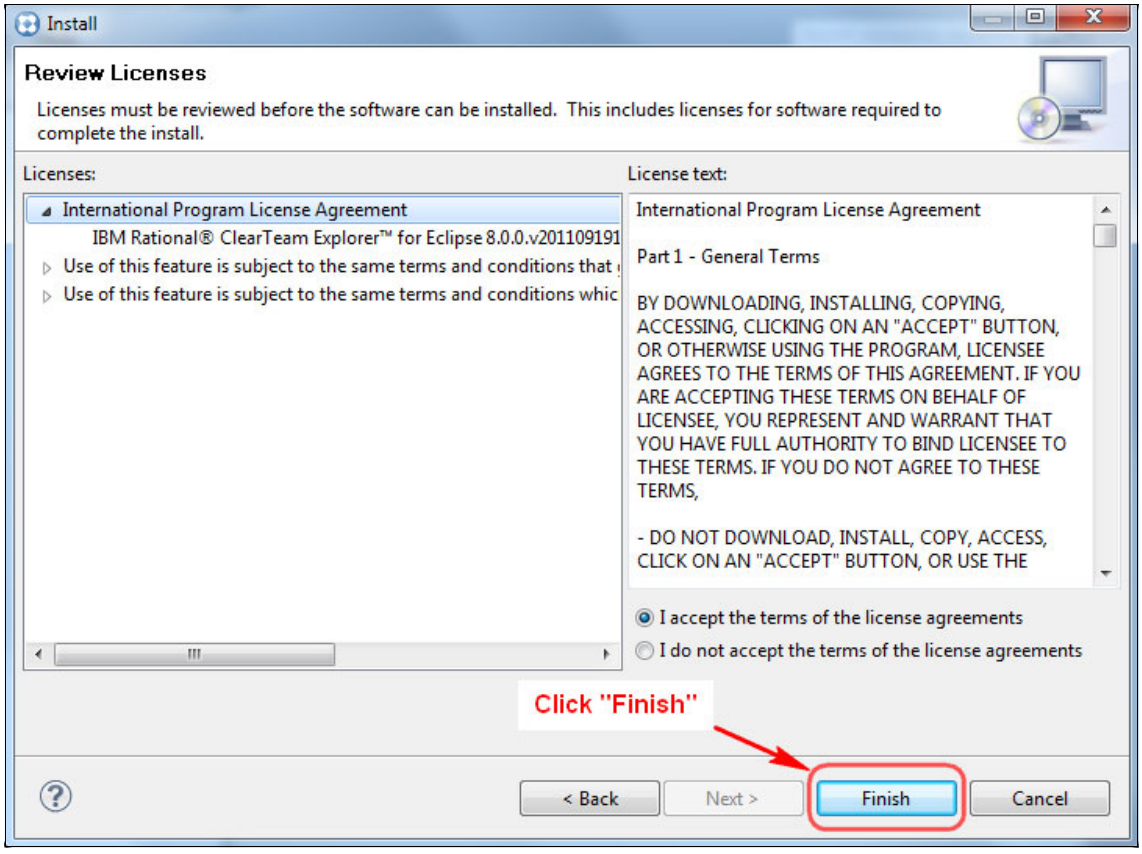


図 4-27　ライセンス条項に同意して「Finish」をクリック

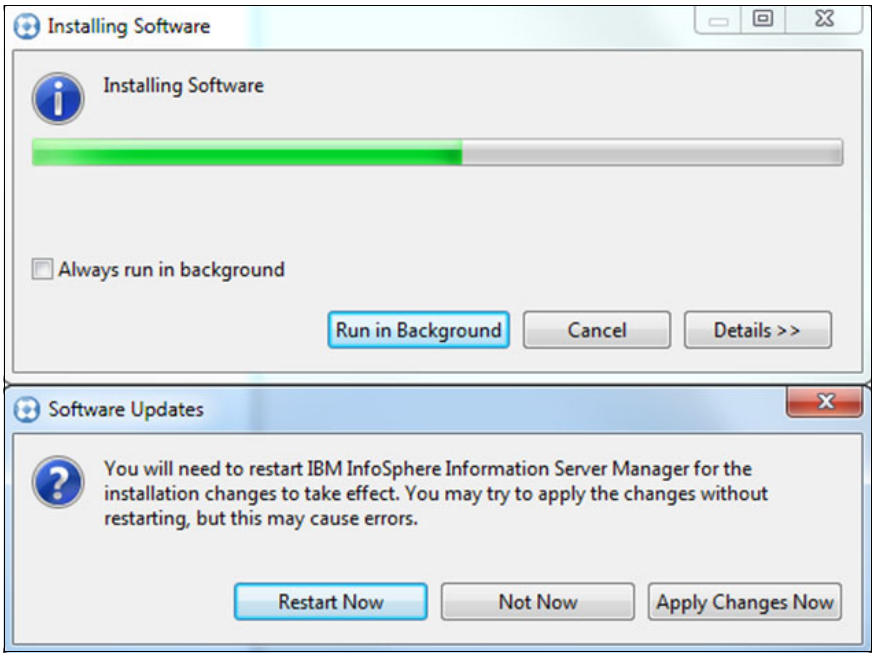


図 4-28　IS Manager の再始動

4.14 ソース管理の統合

このセクションのステップは、ドメインで 1 回のみ実行する必要があります。その結果、ClearCase サーバーのソース管理リポジトリー内に資産が作成され、管理者がこれらのステップを実行する Windows マシンにローカル・ワークスペースも作成されます。

図 4-29 (93 ページ) から図 4-35 (98 ページ) には、ソース管理システムを IS Manager に統合するための以下の手順が示されています。

1. ドメイン名を右クリックして「**Integrate Source Control**」を選択することにより、統合プロセスを開始します。「**Share project**」チェック・ボックスが選択されていることを確認して、「**OK**」をクリックします。
2. 次のようにして ClearCase ビューを作成します。
   a. プロジェクトを共有するためのプラグインを選択して、「**Next**」をクリックしてから、「**Create View**」をクリックします。
   b. 「**Create a base ClearCase view**」を選択して、「**Next**」をクリックし、ユーザー資格情報を指定して、「**OK**」をクリックします。
   c. 「**Next**」をクリックして、ビューのタグ名とパスを指定し、「**Finish**」をクリックします。
3. プロジェクト・フォルダーのロケーションを選択します。
4. 「**OK**」をクリックします。
5. 新しいソース管理ワークスペースを参照します。

その他の開発者およびリリース管理者はソース管理ワークスペースをインポートする必要があります。この手順については、4.15『ソース管理プロジェクトのインポート』（98 ページ）で説明しています。

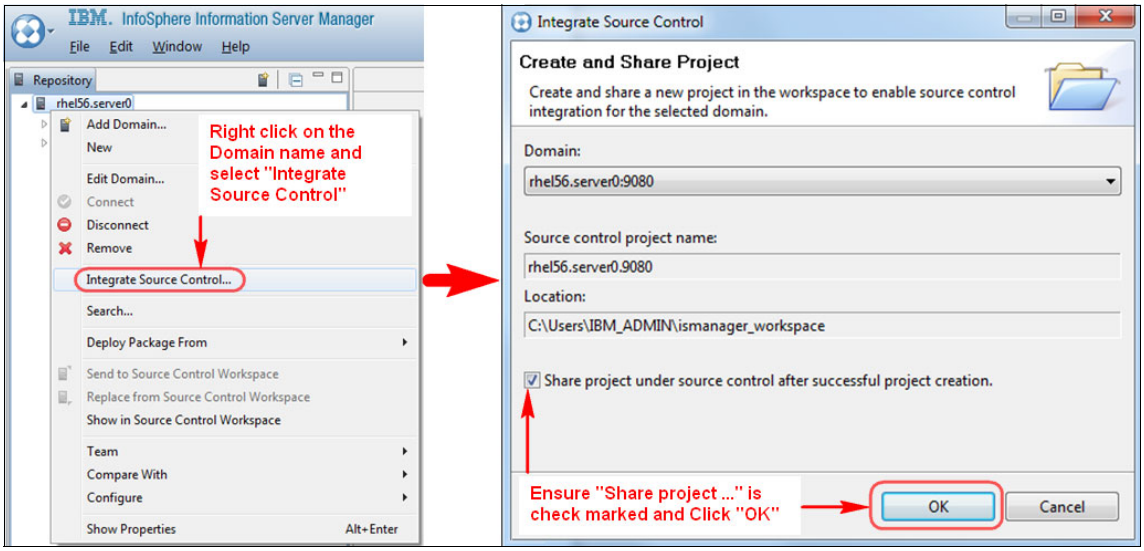


図 4-29　ソース管理統合プロセスの開始

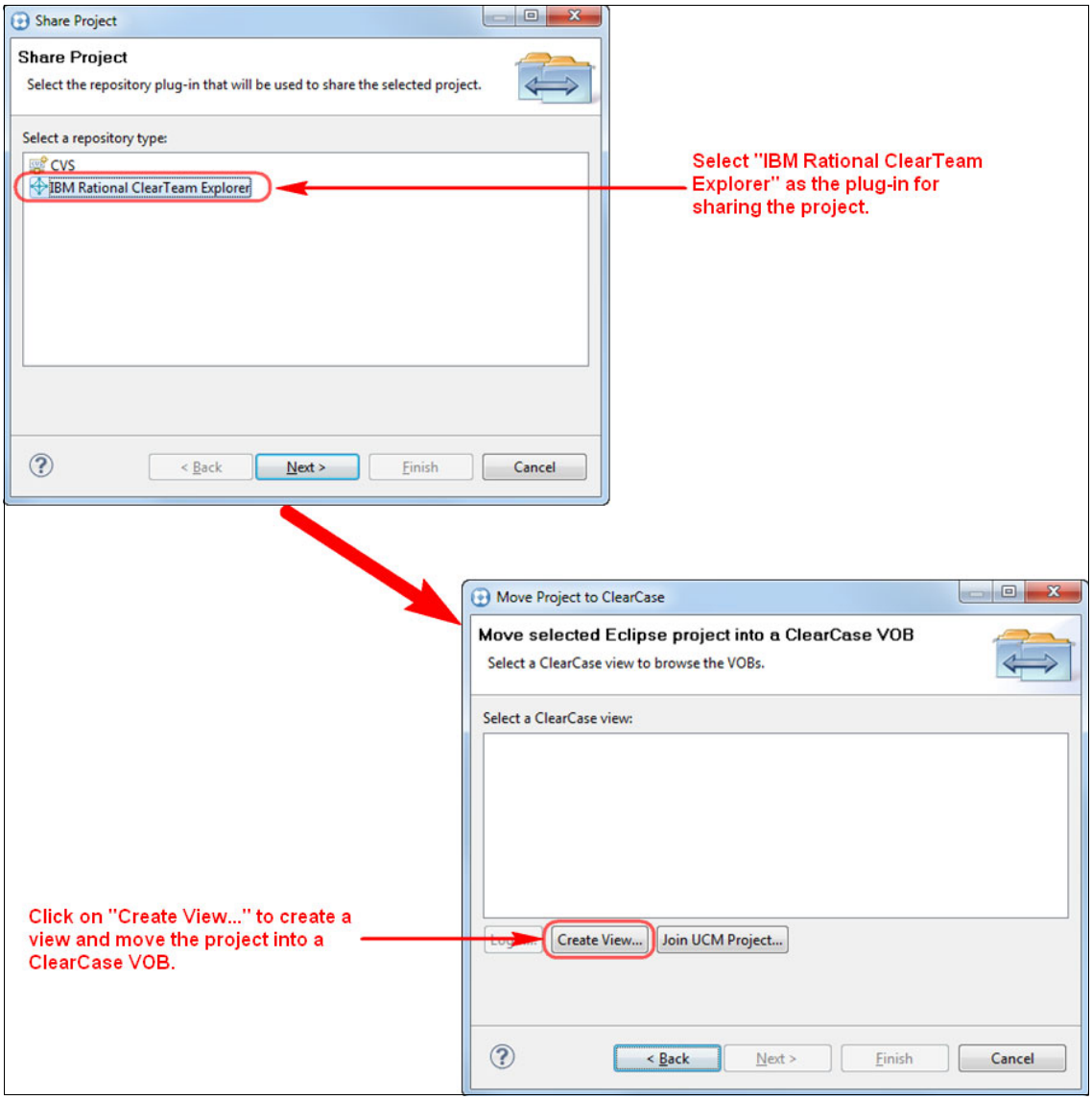


図 4-30　ClearCase ビューの作成 (1/3)

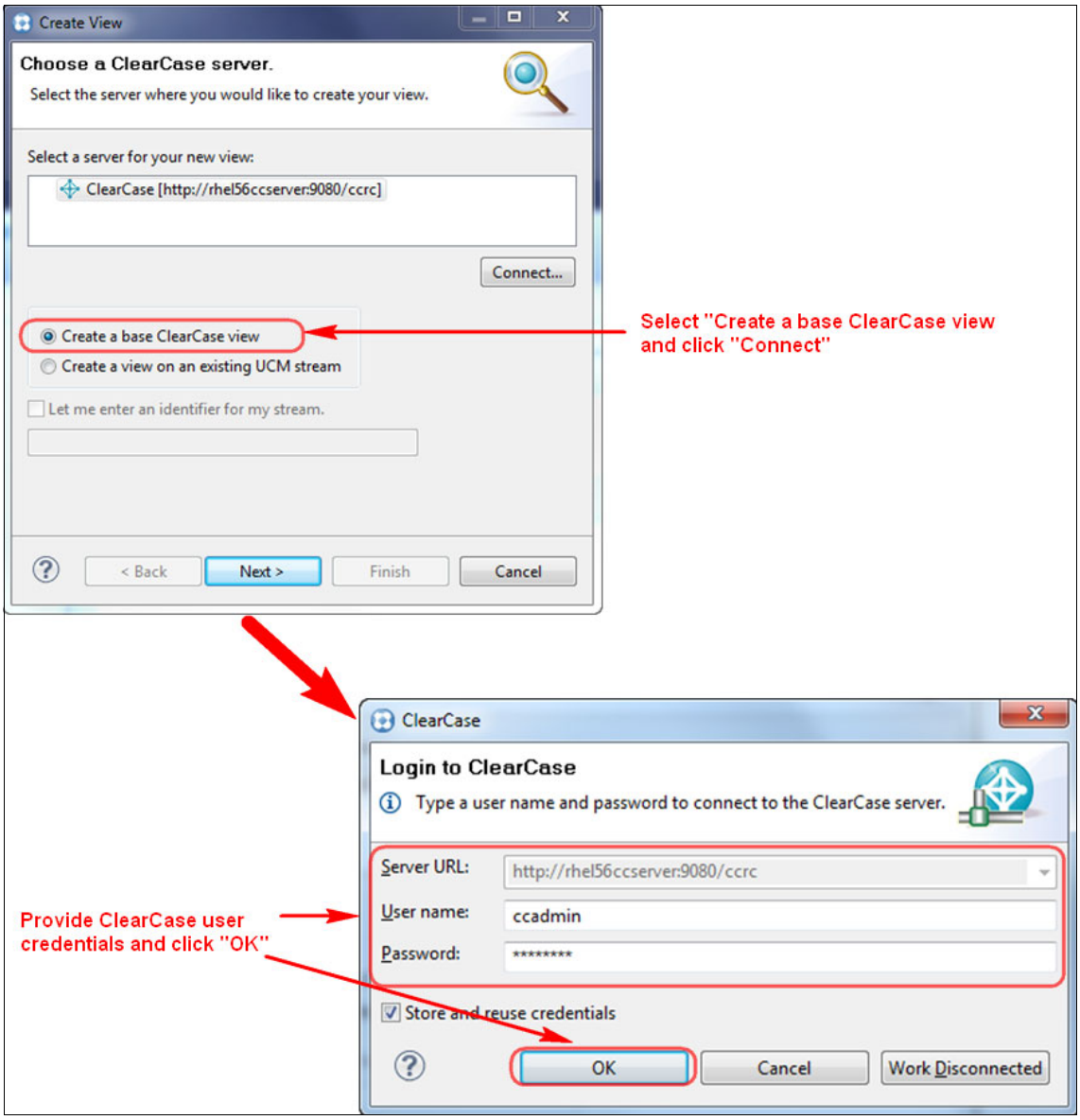


図 4-31　ClearCase ビューの作成 (2/3)

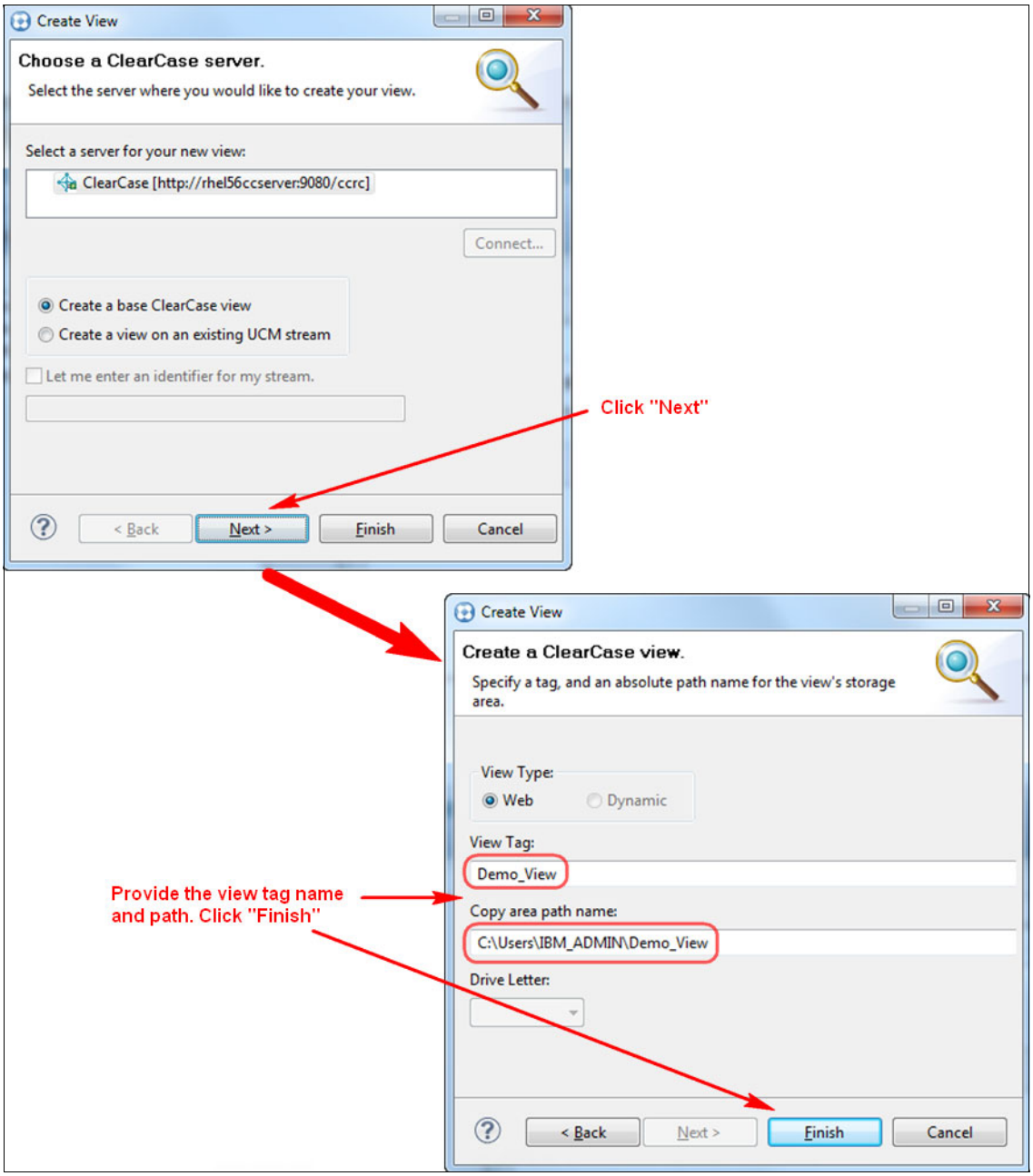


図 4-32　ClearCase ビューの作成 (3/3)

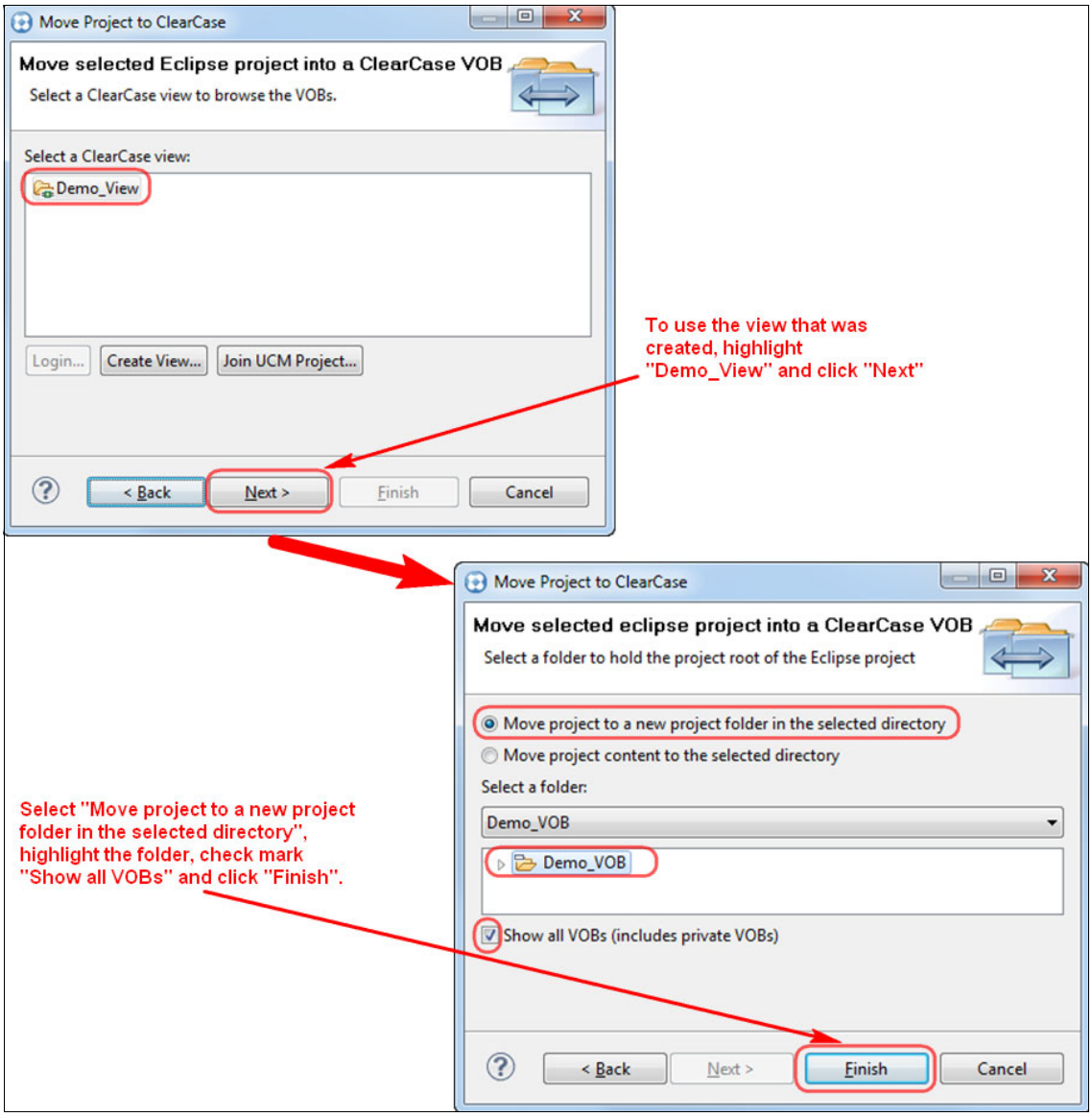


図 4-33　プロジェクト・フォルダーのロケーションの選択

図 4-34 は、セットアップの終了方法を示しています。

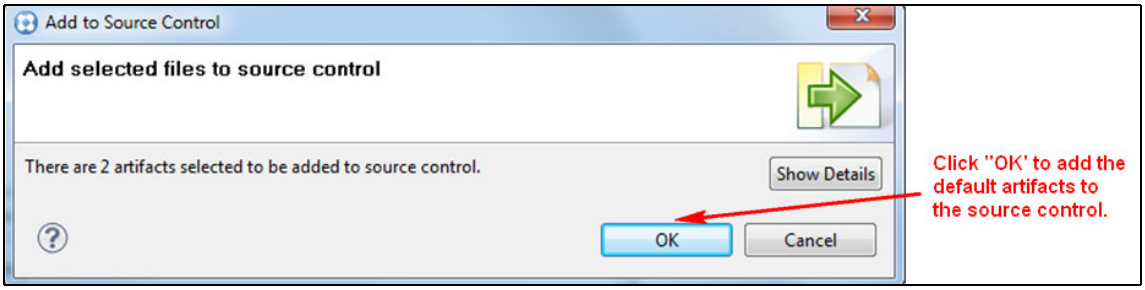


図 4-34　ソース制御統合プロセスのセットアップの終了

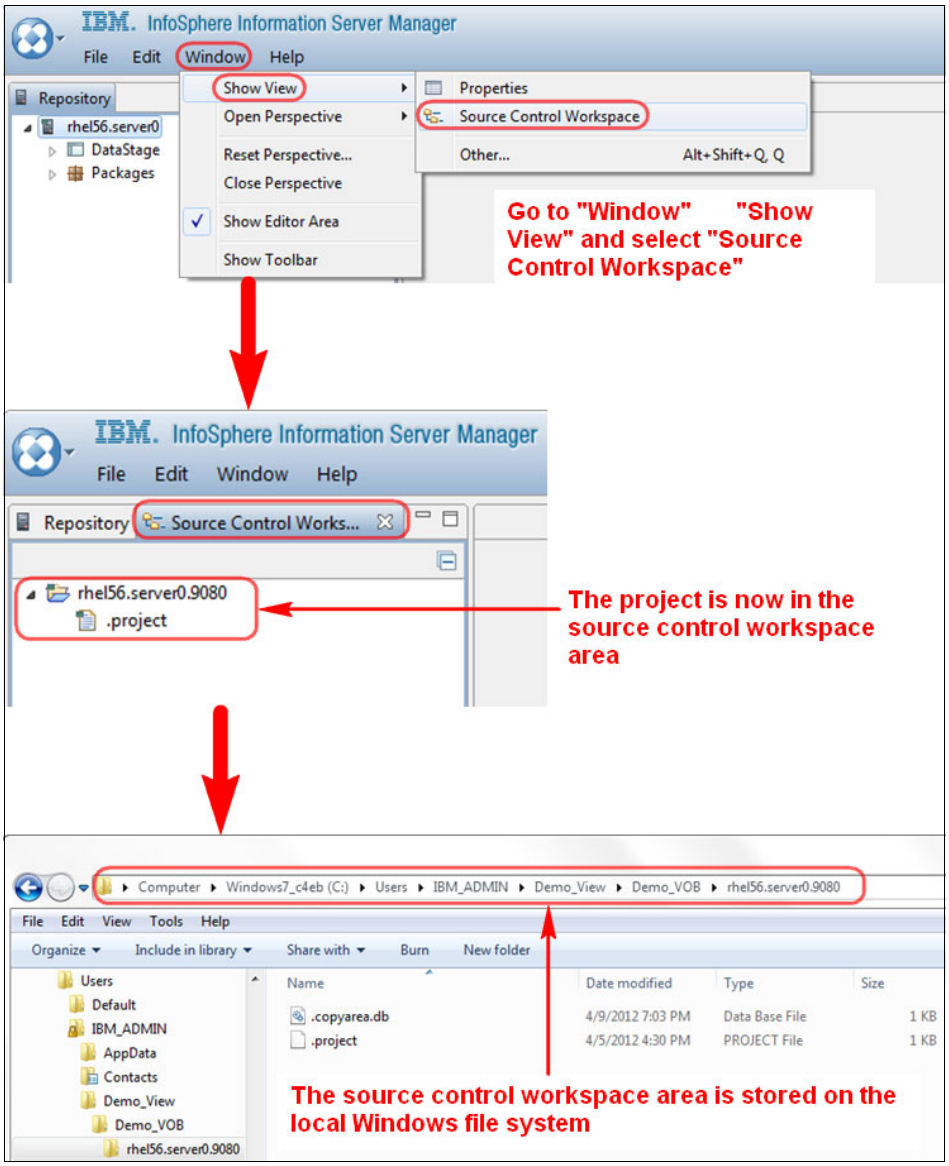


図 4-35　IS Manager およびローカル Windows ファイル・システムでの新しく作成されたソース制御ワークスペースの参照

4.15　ソース管理プロジェクトのインポート

その他すべての開発者およびリリース管理者は、4.14『ソース管理の統合』（93 ページ）で作成されたソース管理プロジェクトをインポートする必要があります。その結果、各ユーザーの Windows クライアント・ワークステーションにローカル・エリアが作成されます。このプロセスでは、ファイルおよびディレクトリーを ClearCase リポジトリーからローカル・ワークスペース・ディレクトリーにもってきます。

最初に 4.14『ソース管理の統合』（93 ページ）の手順を完了していることを確認してください。このセクションの図 (図 4-36 から図 4-42 (105 ページ)) には、その他のユーザーのために既存のソース管理プロジェクトをインポートする以下の手順が示されています。

1. 「**Window**」→「**Show View**」→「**Source Control Workspace**」を選択して、ワークスペースを表示します。
2. 「**Window**」→「**Open Perspective**」→「**Other**」を選択します。「Open Perspective」ダイアログで選択して、「**OK**」をクリックします。

3. 「**My Views**」を右クリックして、「**Connect**」を選択します。接続情報の詳細を入力して、「**OK**」をクリックします。
4. 「**My Views**」を右クリックして、「**Create View**」を選択し、「**Create a base ClearCase view**」を選択して、「**Next**」をクリックすることにより、ビューを作成します。
5. ビューのリソースを選択します。
6. IS ドメイン・フォルダーをローカル・ワークスペースにインポートします。
7. 新しく作成されたワークスペースで、インポートされたプロジェクトを参照します。

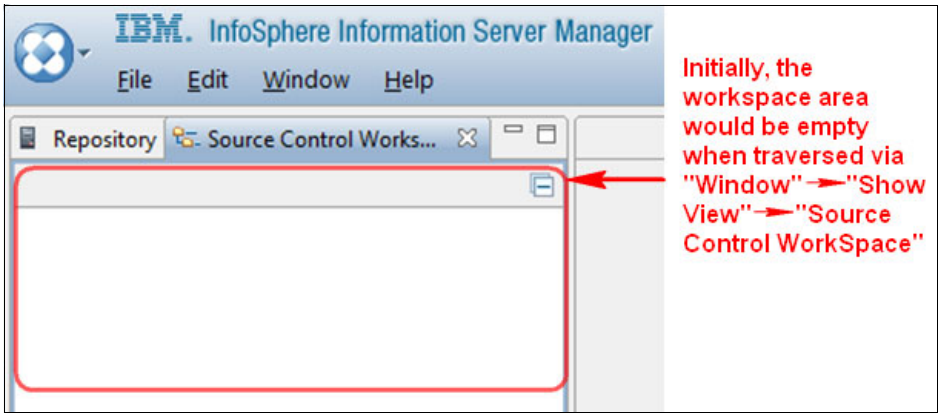


図 4-36　その他の初回ユーザーの場合、「Source Control Workspace」エリアは空

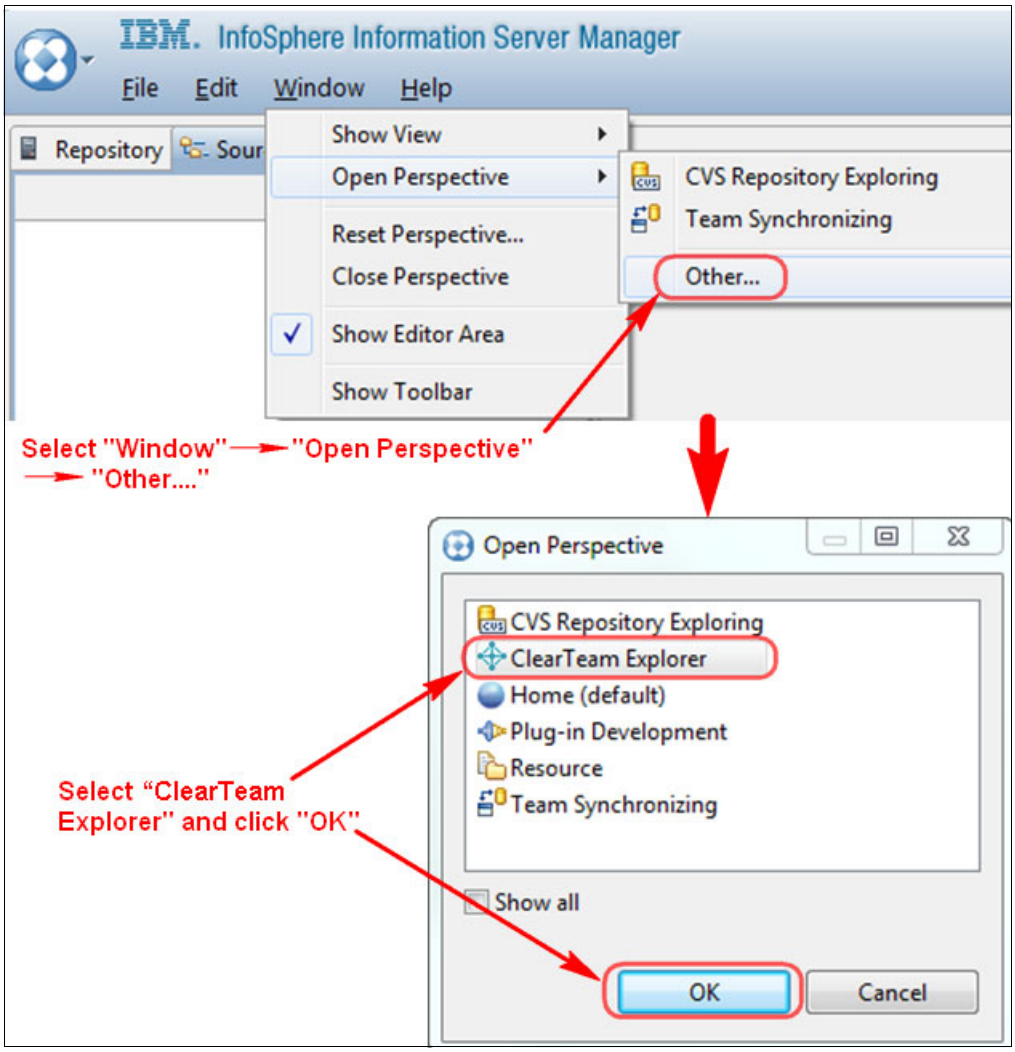


図 4-37　その他の初回ユーザーのための最初のステップ

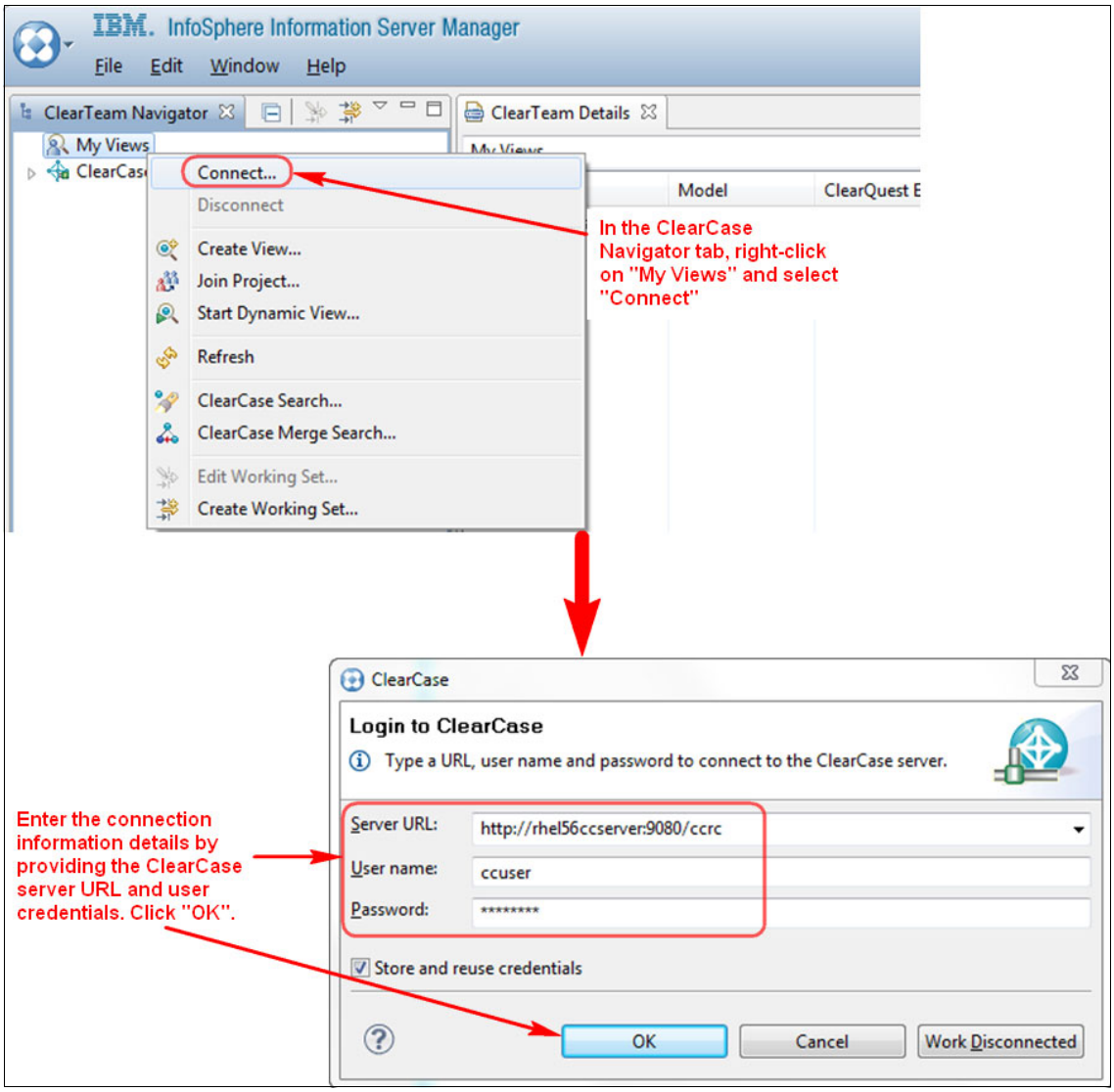


図 4-38　ClearTeam Navigator 内からの ClearCase サーバーへの接続

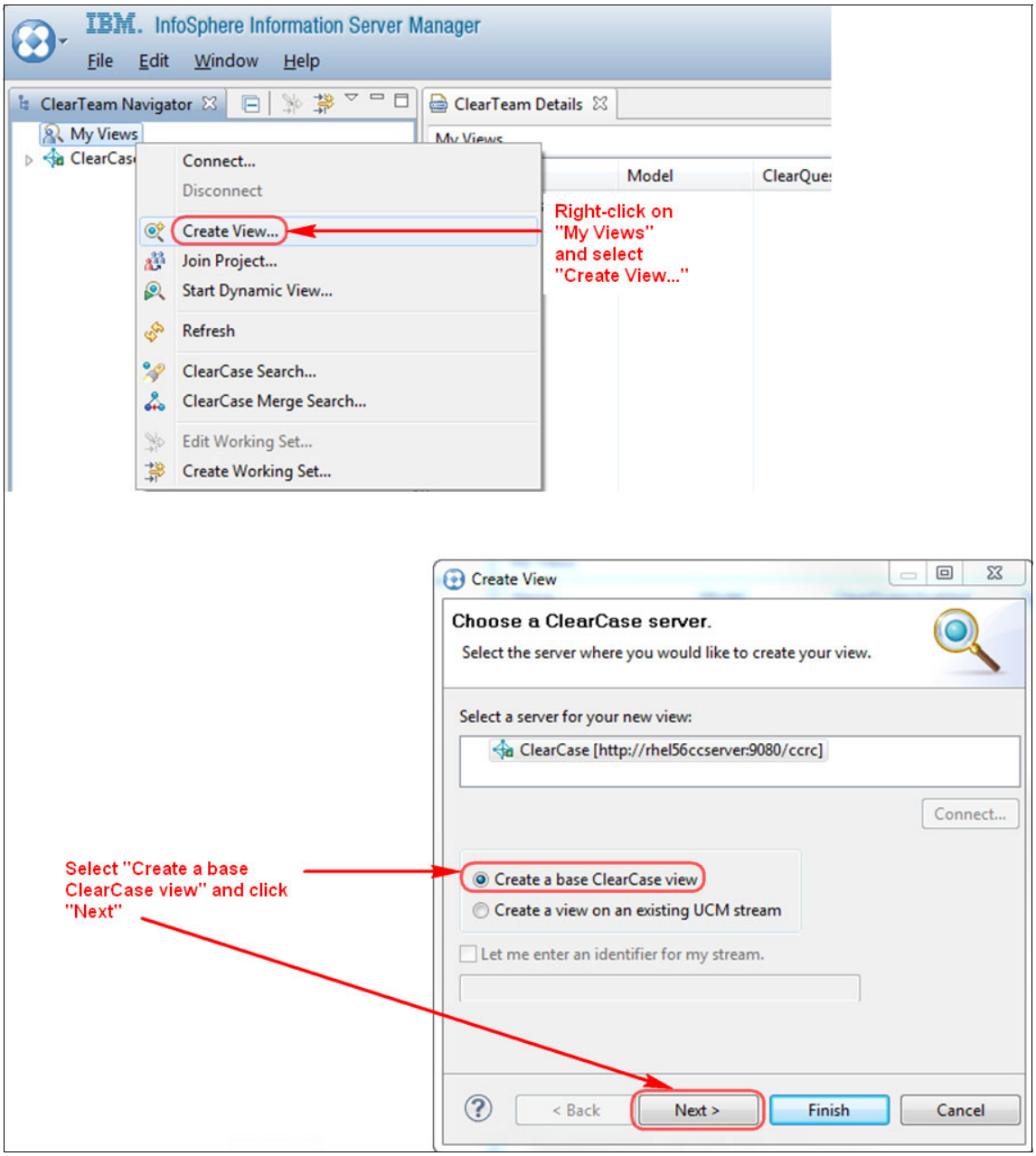


図 4-39　ClearTeam Navigator を使用した基本の ClearCase ビューの作成

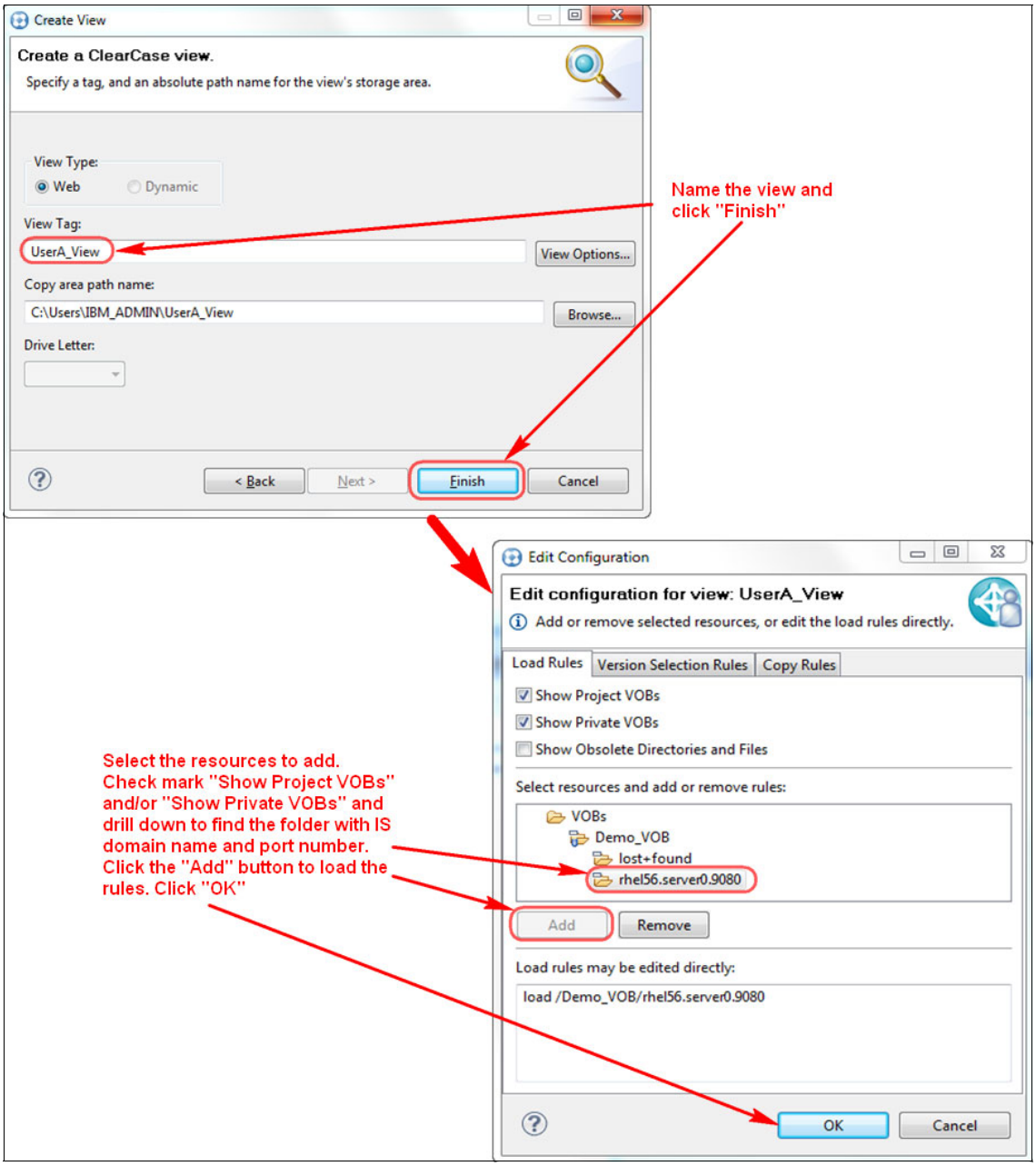


図 4-40　ビューのリソースの選択

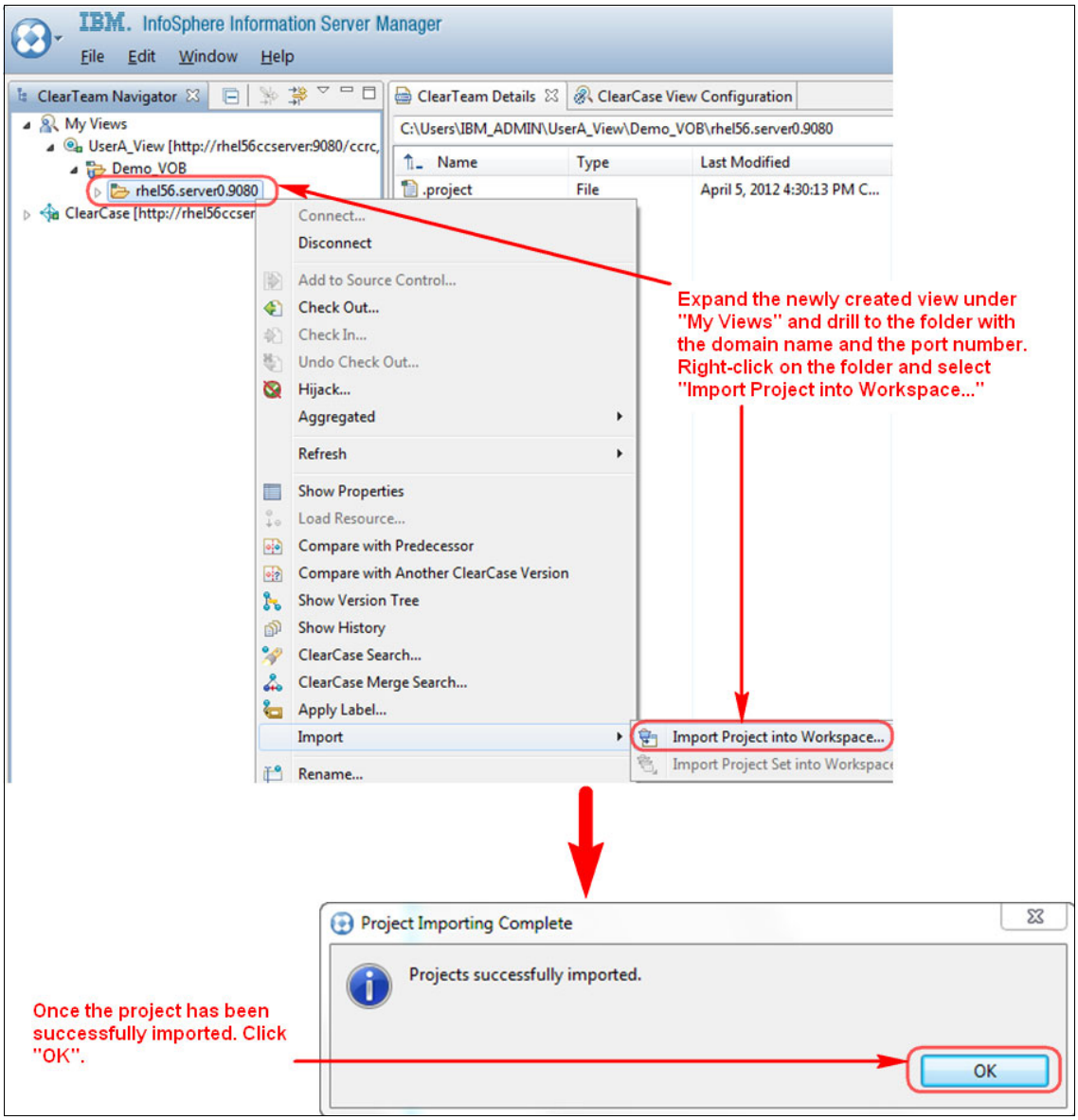


図 4-41　ローカル・ワークスペースへの IS ドメイン・フォルダーのインポート

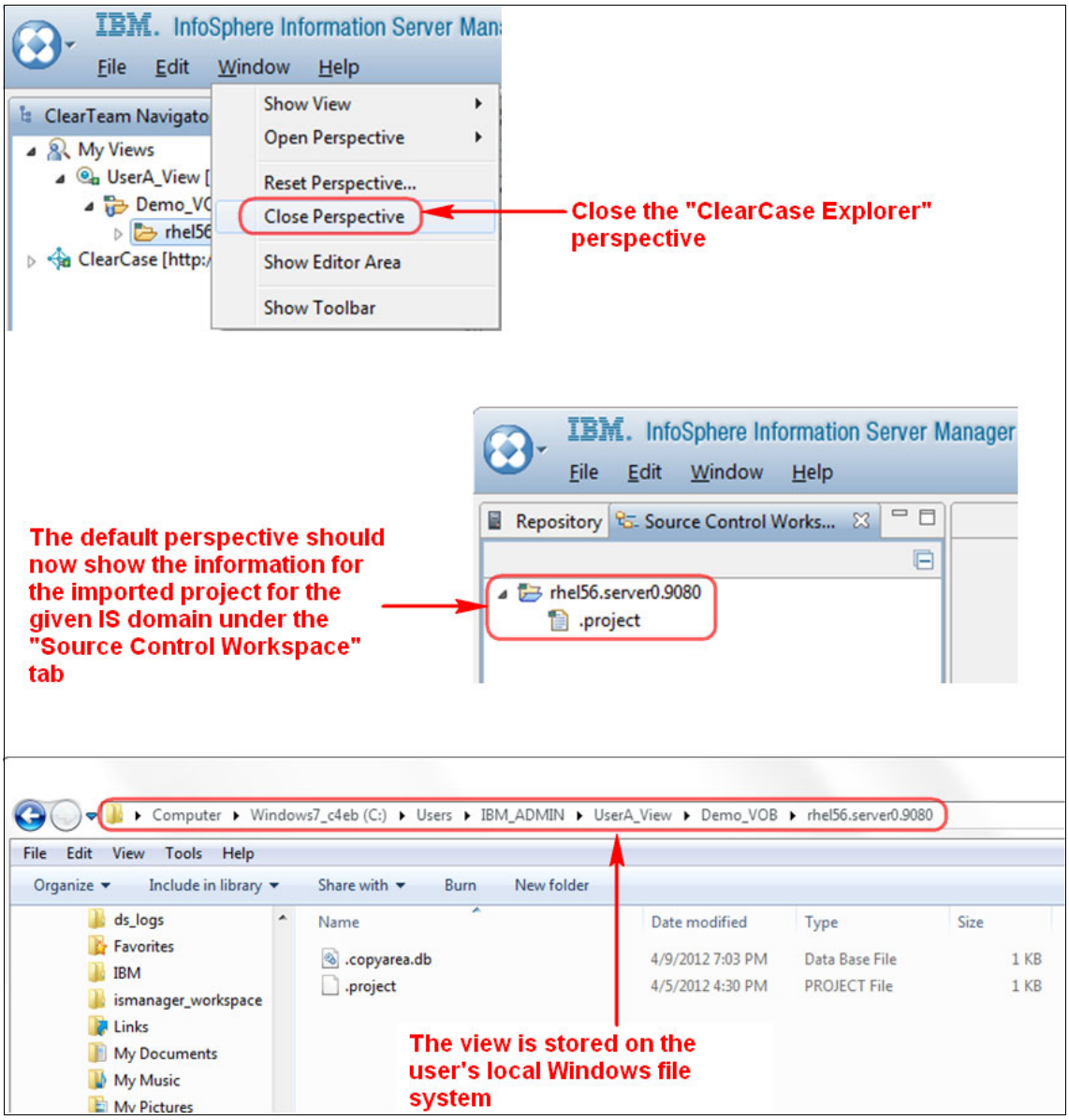


図 4-42　新しく作成されたワークスペースでのインポートされたプロジェクトの参照

4.16　「Source Control Workspace」への資産の追加

このセクションでは、新しい DataStage 資産 (ジョブやジョブ・シーケンスなど) を Information Server リポジトリーから「Source Control Workspace」に追加する方法を説明します。

「Source Control Workspace」に追加することにより、資産は黙示的に ClearCase リポジトリーにもアップロードされます。図 4-43 (106 ページ) を参照してください。

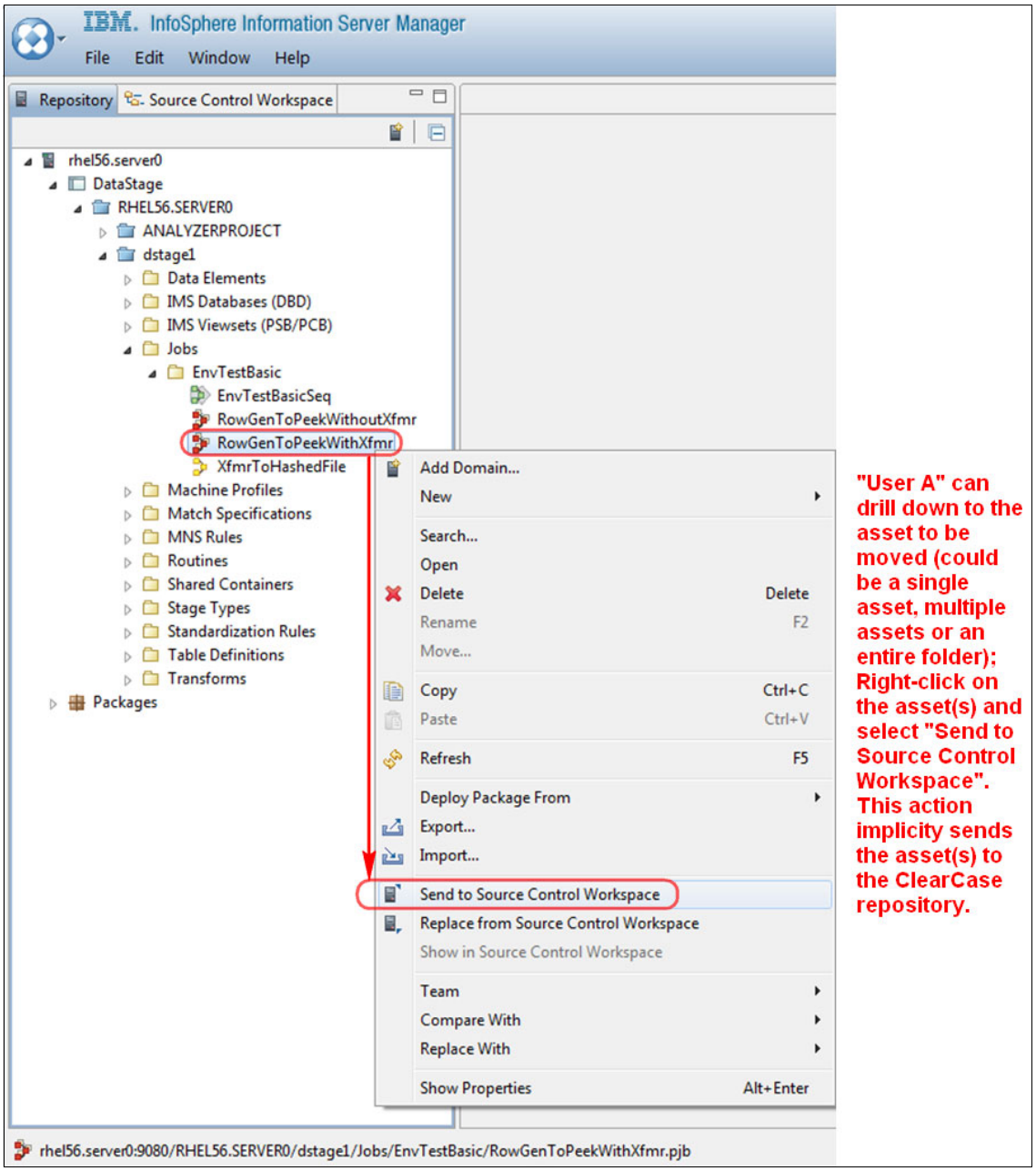


図 4-43　「Source Control Workspace」への資産の移動

図 4-44 では、ラベル / アイコン装飾機能を強調表示しています。大半のソース管理システムおよびツールには、独自のバージョンのラベル / アイコン装飾機能があります。例えば、CVS ではラベル / アイコン装飾機能の表現が異なります。本書ではソース管理システムとして ClearCase を使用しているため、ClearCase のラベル / アイコン装飾機能を使用します。

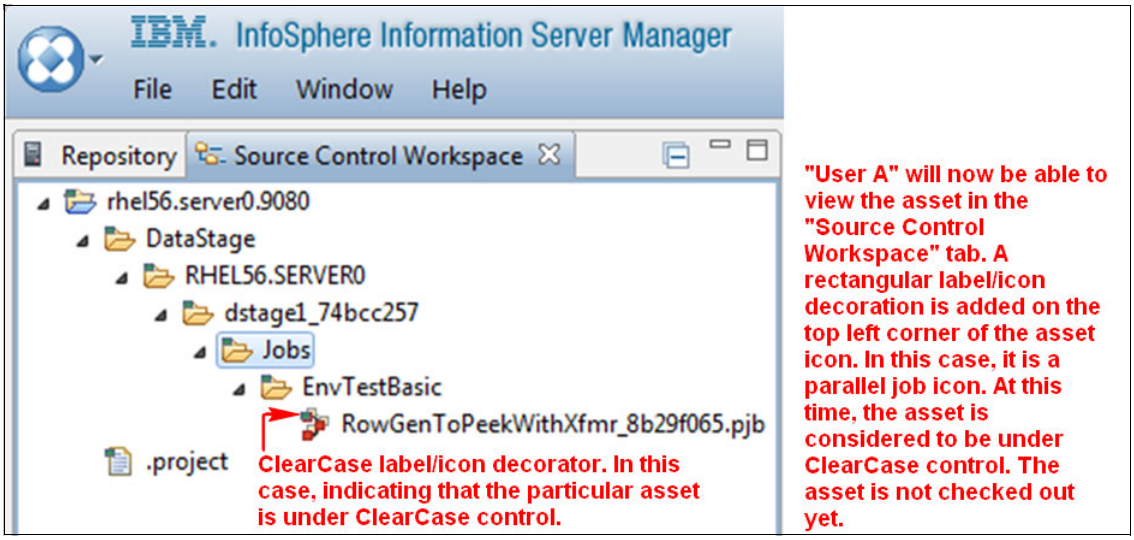


図 4-44 「Source Control Workspace」での資産の表示

ClearCase の用語では、*ラベル装飾機能* とは、ClearCase ビューでリソースの状態を示し、資産のラベルおよびアイコンに追加情報を表示できるようにするものです。図 4-45 および図 4-46 (108 ページ) に、本書で取り上げている ClearCase のバージョンで使用されているラベル装飾機能を示します。特定のソース制御システムのバージョン間でラベル / アイコン装飾機能が異なる場合があります。

ClearCase state	Icon decoration	Text decoration	Resource type	Description
Under ClearCase control		@@	File or directory	The resource is under ClearCase control and has been loaded into the view's copy area.
View private		none	File or directory	The resource is an ordinary file or directory in the local file system. It is not under ClearCase control.
Partially loaded		@@PL	directory	The directory is loaded but contains one or more resources that have not been loaded into the view's copy area.
Not loaded		@@NL	File or directory	The resource is under ClearCase control but has not been loaded because it is not selected by the ClearCase view's load rules.
Upgrade view		none	View	The view was created in a version of ClearCase prior to 7.1. The view needs to be upgraded if it is to be used in 7.1. Note that if you upgrade the view to 7.1, it will no longer be functional or visible in prior versions of ClearCase.
Checked out		@@CO	File or directory	The resource is checked out.
Aggregated change		none	File or directory	At some point at or below the current node level, a resource has been changed (checked out or hijacked).

図 4-45　ClearCase のラベル / アイコン装飾機能 (1/2)

Aggregated and checked out		none	Directory	An aggregated directory resource is checked out and may also have checkouts.
Hijacked		@@HJ	File	The file is hijacked.
Unknown		@@?	File or directory	The resource has an unknown ClearCase state.
VOB		none	directory	The resource is a ClearCase VOB.
UCM resource		none	View, file, or directory	The resource is part of a ClearCase UCM project.
Discordant state		none	File or directory	The resource is in a discordant state.
Symbolic link		none	File or directory	The resource is a symbolic link.
Component VOB		none	VOB	The resource is a Rational ClearCase component VOB.
Project VOB		none	VOB	The resource is Rational ClearCase project VOB.
ClearQuest project		none	ClearCase project	The resource is a Rational ClearCase UCM project that is integrated with Rational ClearQuest.
Obsolete		none	Stream, project, VOB, activity	The resource has been designated as Obsolete through the ClearCase Properties view.
Locked		none	Stream, project, VOB, activity	The resource has been designated as Locked through the ClearCase Properties view.
Current activity		none	Activity	The resource is the current activity.

図 4-46　ClearCase のラベル / アイコン装飾機能 (2/2)

図 4-47 は、「Team」メニューを使用して ClearCase コマンドにアクセスする手順を示しています。「Team」メニューからすべてのソース管理システムのコマンドを使用できます。

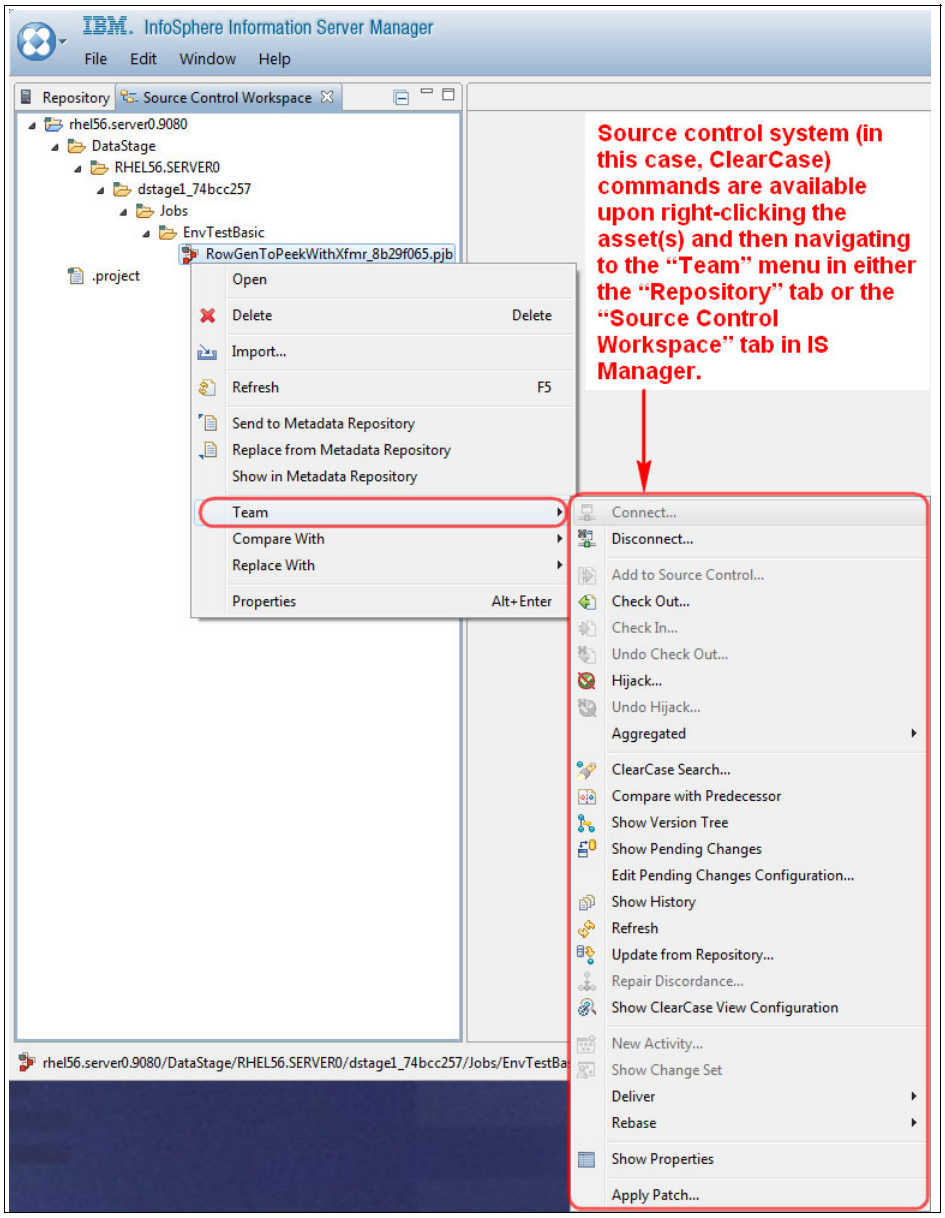


図 4-47　「Team」メニューを使用した ClearCase コマンドへのアクセス

図 4-48 は、「Team」メニューを使用して ClearCase リポジトリー内の資産のバージョン・ツリーを表示する手順を示しています。

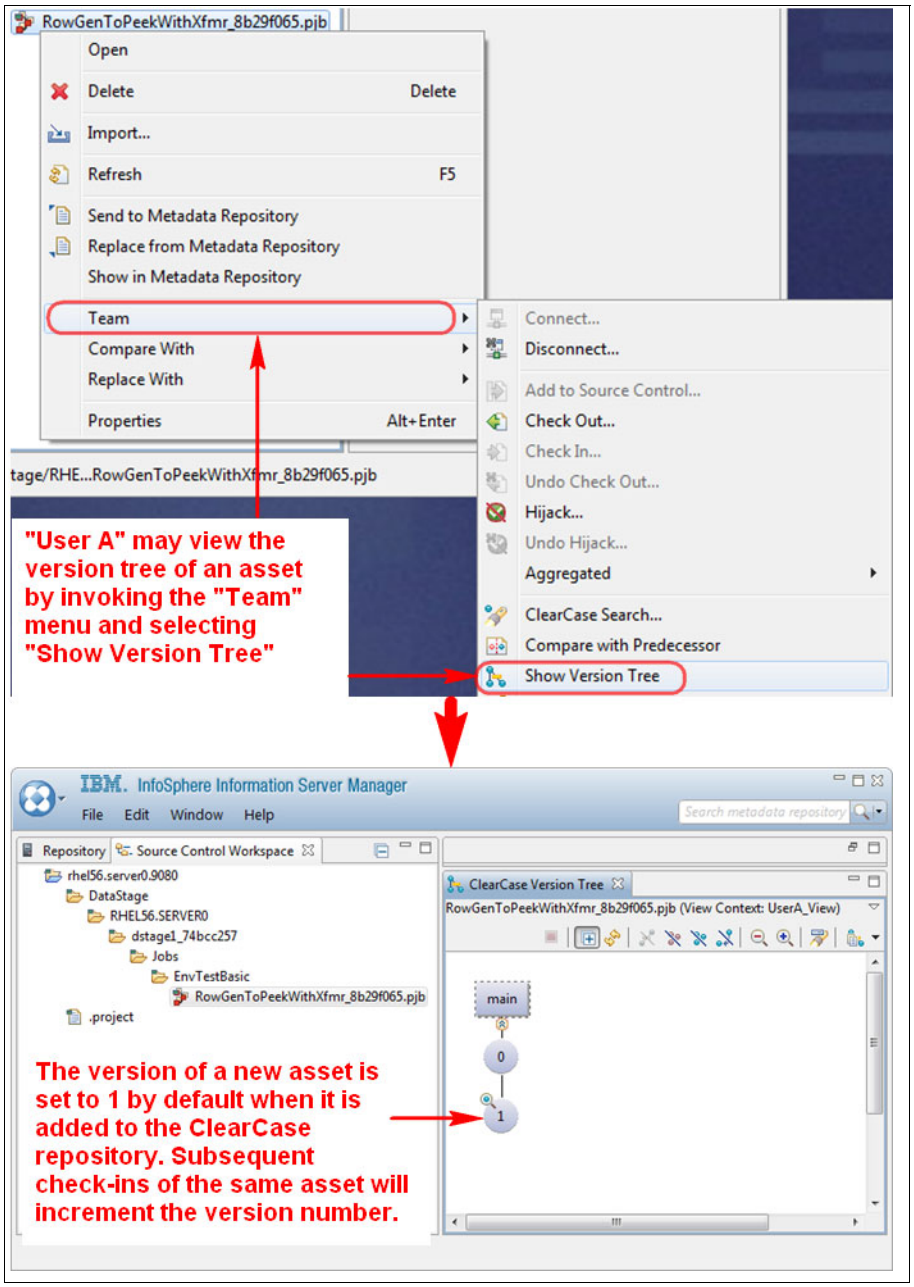


図 4-48　資産のバージョン・ツリーの表示

4.17 チェックアウト

このセクションでは、既にソース管理リポジトリーに追加されている資産をチェックアウトするステップを説明します。

図 4-49 (111 ページ) は、チェックアウト後に資産のラベル / アイコン装飾機能がどのように変化するかを含めて、チェックアウト手順を示しています。

チェックアウト・オプションにより、ClearCase サーバーのソース管理リポジトリーからローカルのソース管理ワークスペースに資産をもってきます。Information Server リポジトリーが暗黙的に更新されることはありません。

このセクションの図 (図 4-50 (112 ページ) から図 4-53 (114 ページ)) には、チェックアウトされたばかりの資産によって Information Server リポジトリーを更新する 2 つの方法が示されています。「Repository」または「Workspace」のどちらのビューも使用できます。

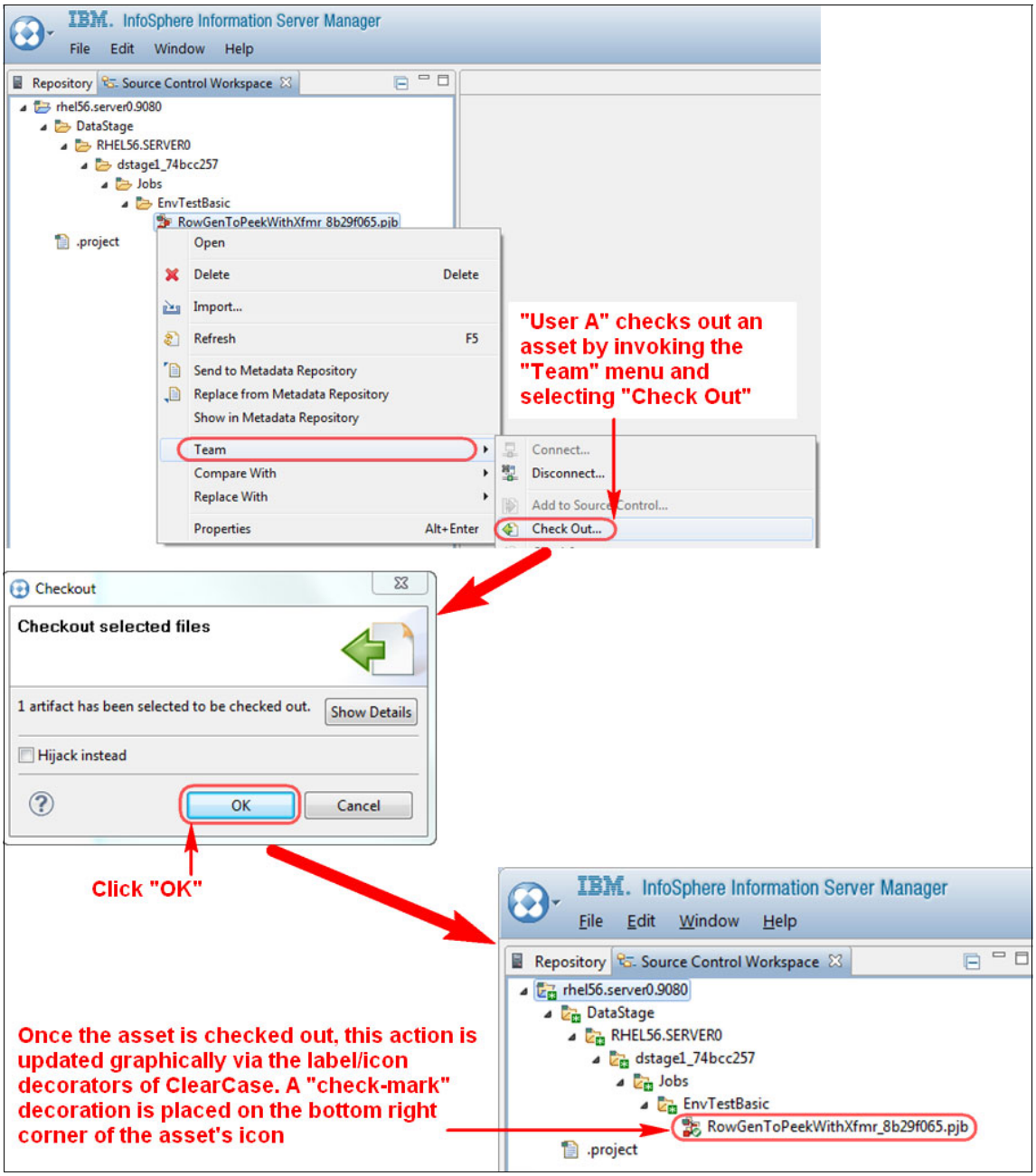


図 4-49　IS Manager の「Workspace」ビューの「Team」メニューを使用した資産のチェックアウト

図 4-50 に示すように、ユーザーは、チェックアウト・プロセス中に「Checkout selected files」(図 4-49 (111 ページ) の中間ステップ) ダイアログの「**Show Details**」をクリックすることにより、資産のチェックアウト中に追加のオプションを指定できます。標準的な手法では、ソース管理システムでの資産のチェックアウト時およびチェックイン時にコメントを指定します。

図 4-50 で強調表示されているように、ClearCase では、ユーザーはチェックアウトのタイプを選択できます。選択されたチェックアウトのタイプは、ClearCase リポジトリー内の資産のみに対して有効になり、Information Server リポジトリー内の資産には影響を与えません。

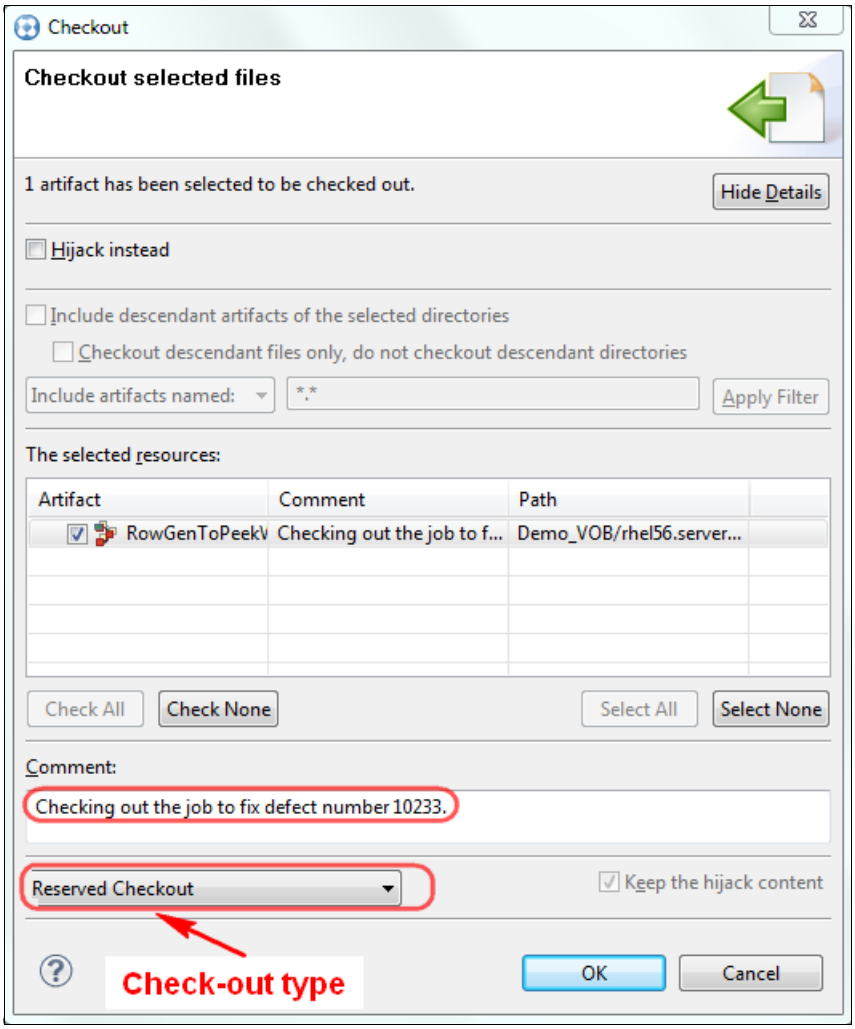


図 4-50　チェックアウト・プロセス中の追加のオプションの指定

図 4-51 および図 4-52 (114 ページ) は、資産がチェックアウトされた後に xmeta リポジトリーを更新する手順を示しています。どちらの方法も使用できます。

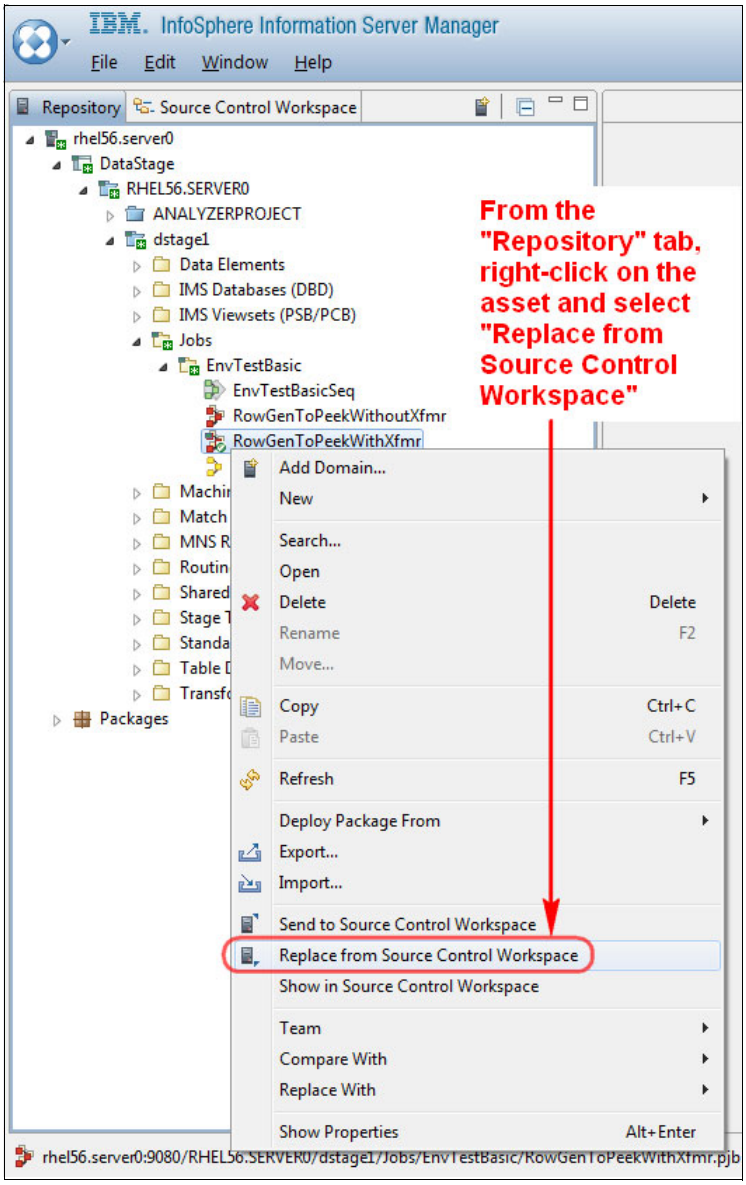


図 4-51　チェックアウト後の xmeta の更新 (「Repository」ビュー)

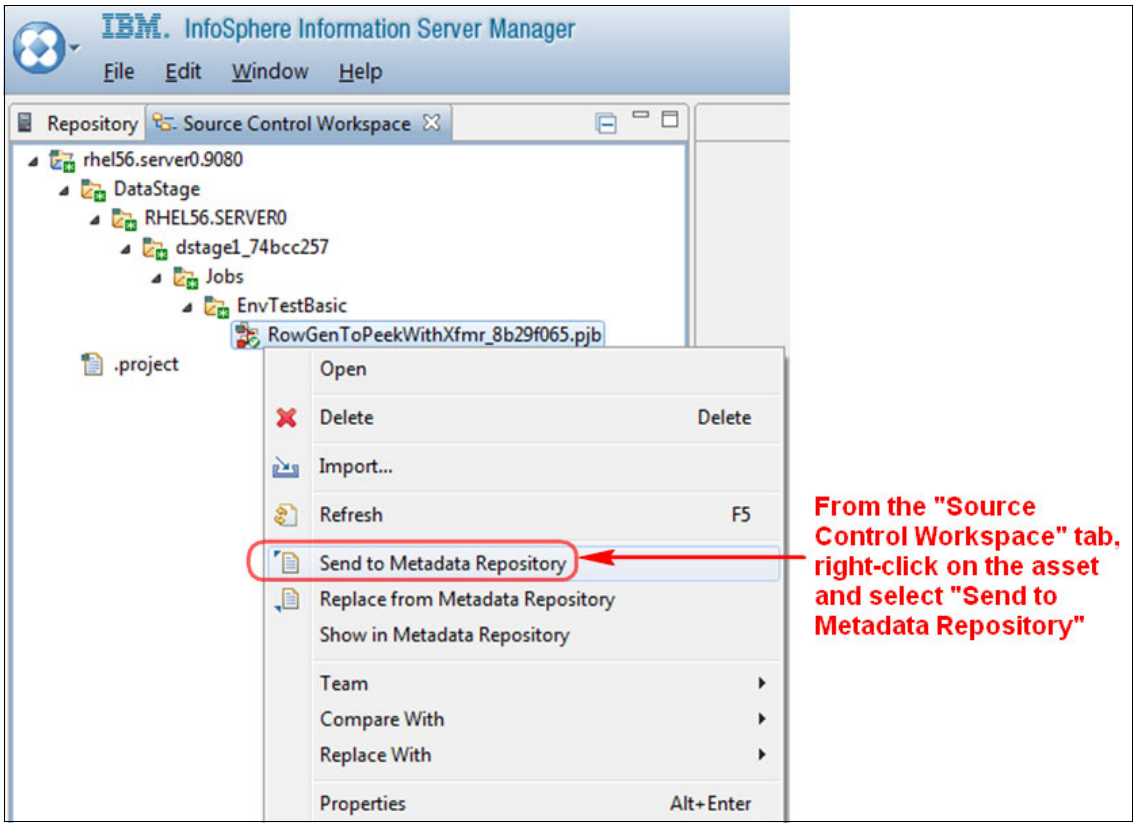


図 4-52　チェックアウト後の xmeta の更新（「Workspace」ビュー）

図 4-53 に示すように、チェックアウトした資産の状況を表示します。

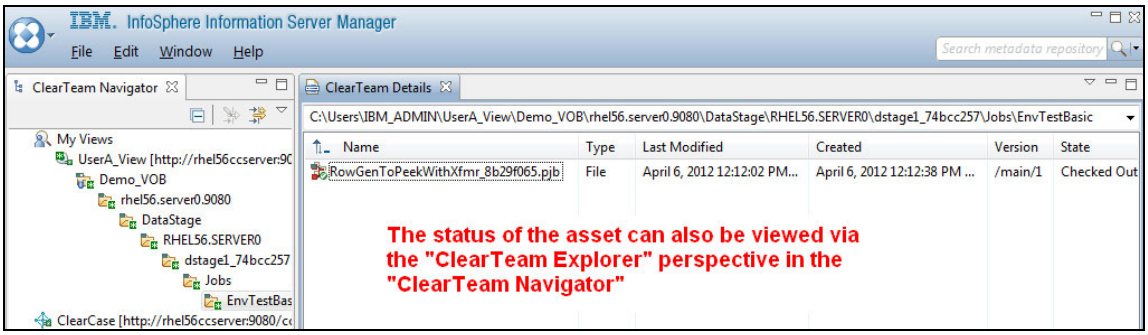


図 4-53　ClearTeam Explorer パースペクティブでのチェックアウトした資産の状況の表示

4.18 チェックイン

このセクションでは、資産を ClearCase に戻してチェックインするためのステップを説明します。

チェックイン・オプション自体によって、DataStage 資産が Information Server リポジトリーから ClearCase にコピーされることはありません。チェックイン・オプションは、ローカルのソース管理ワークスペースに存在するバージョンを移動します。

最初に、Information Server リポジトリーから出した資産によってローカルのソース管理ワークスペースを更新するプロセスについて説明します。「Repository」または「Source Control Workspace」のどちらのビューも使用できます。資産がチェックアウトされた後で DataStage Designer で更新された場合、通常はこの操作を実行します。

図 4-54 (115 ページ) および図 4-55 (116 ページ) は、資産をチェックインする前に「Source Control Workspace」を更新する手順を示しています。どちらの方法も使用できます。

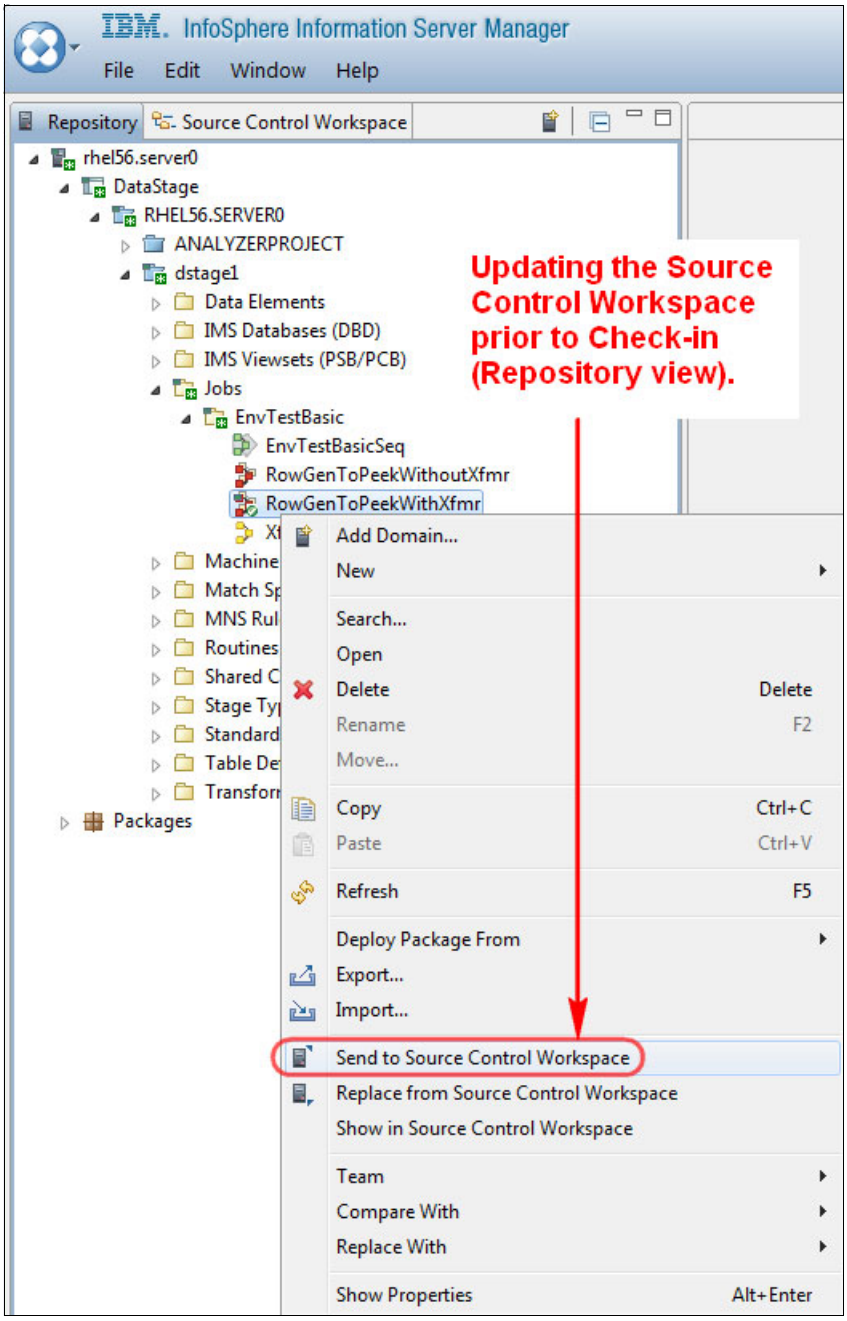


図 4-54　チェックイン前の Source Control Workspace の更新 (「Repository」ビュー)

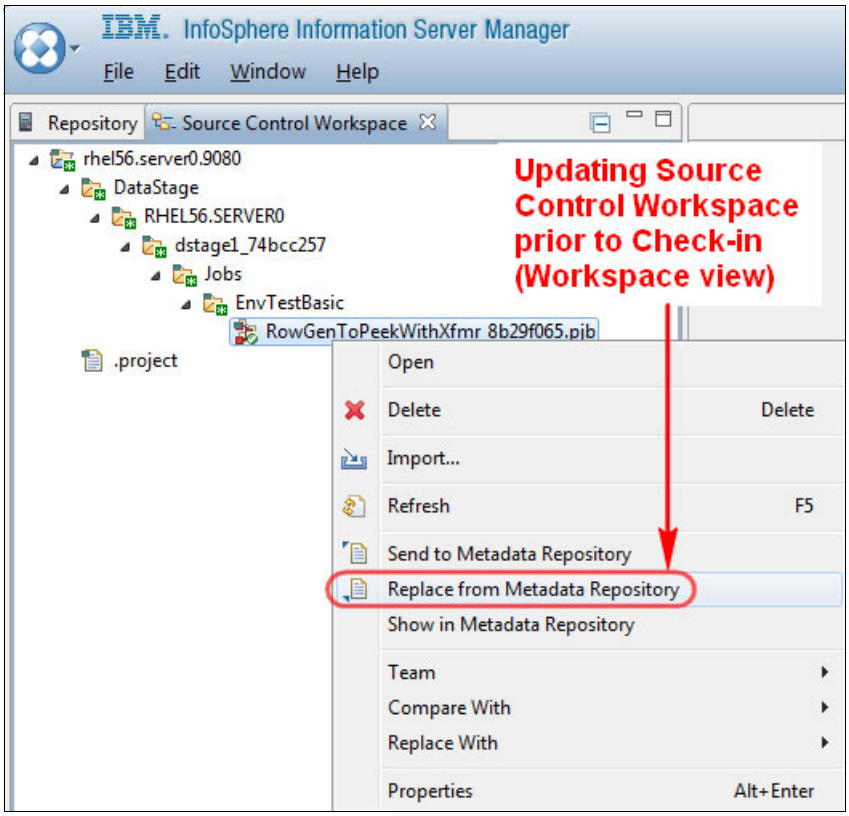


図 4-55　チェックイン前の Source Control Workspace の更新（「Workspace」ビュー）

図 4-56 に、チェックイン・プロセスを実際に実行する方法を示します。

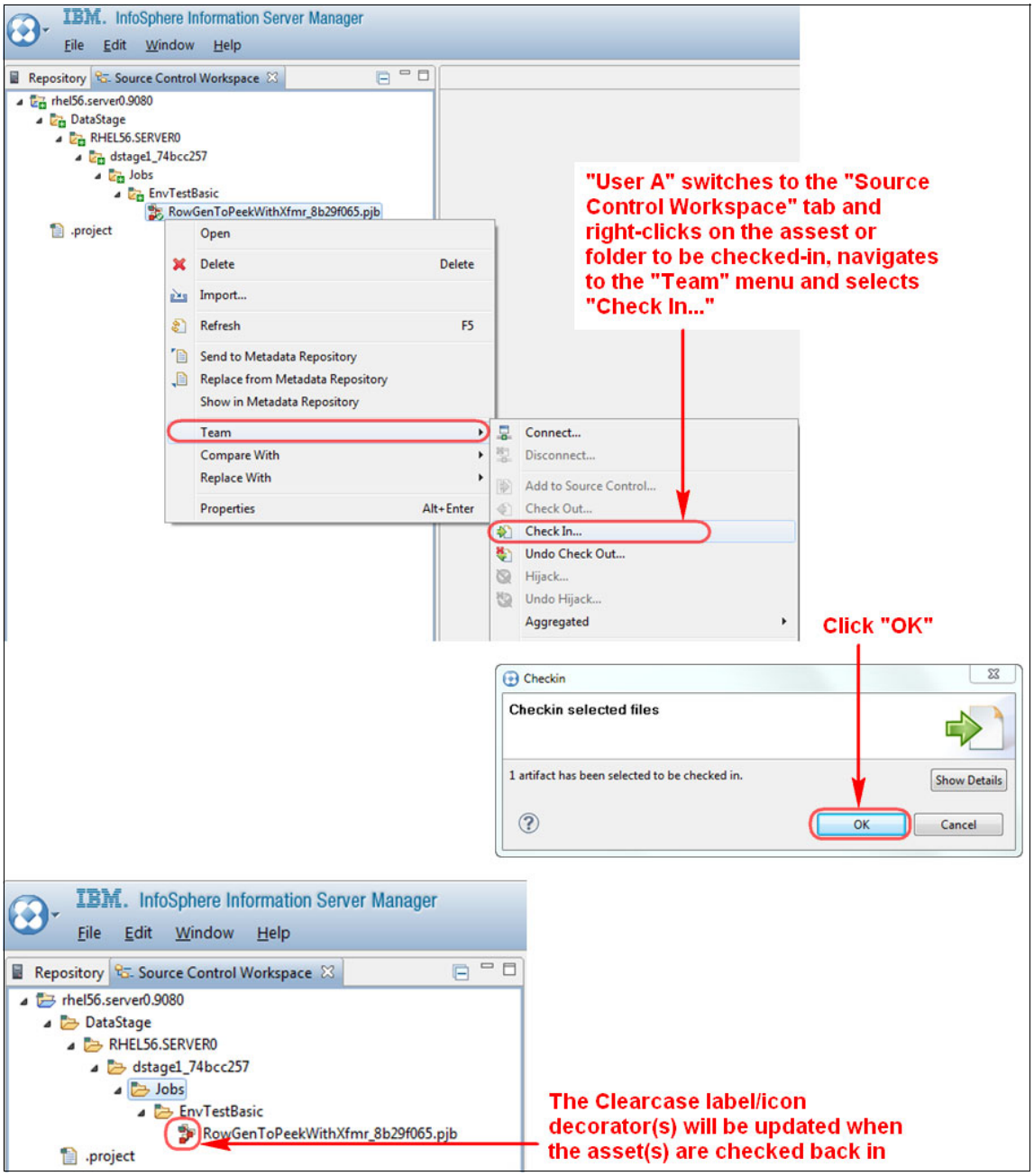


図 4-56　IS Manager の「Source Control Workspace」ビューの「Team」メニューを使用した資産のチェックイン

図 4-57 に示すように、ユーザーは、チェックイン・プロセス中に「**Checkin selected files**」ウィンドウ (図 4-56 (117 ページ) の中間ステップ) の「**Show Details**」をクリックすることにより、資産のチェックイン中に追加のオプションを指定できます。

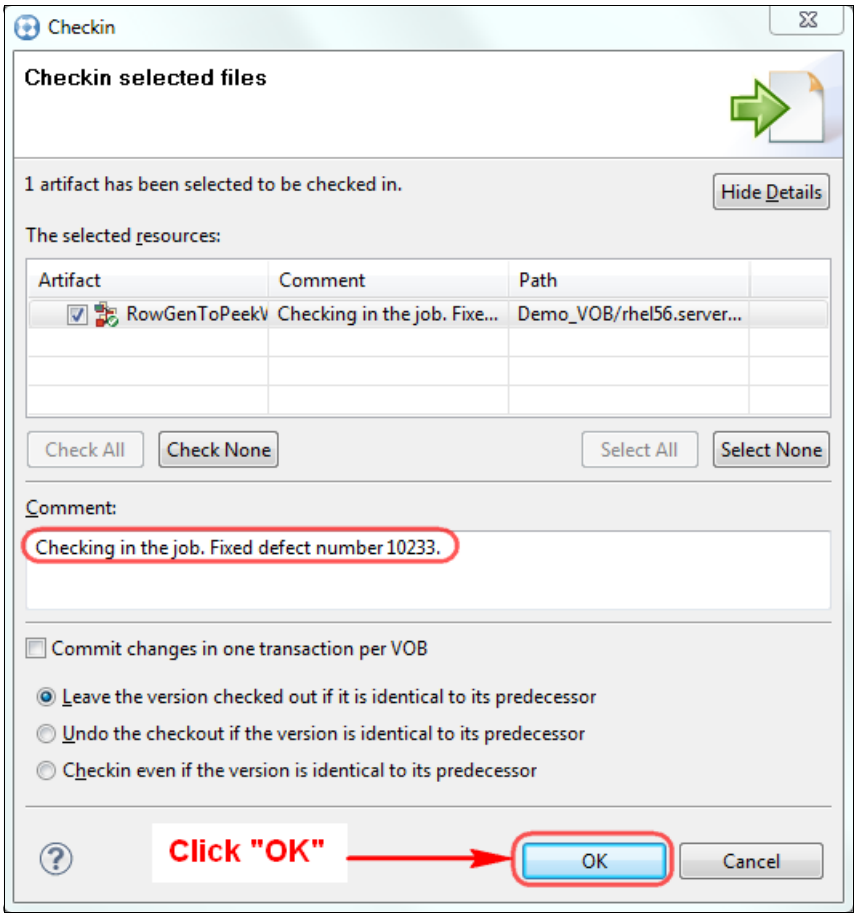


図 4-57　チェックイン・プロセス中の追加のオプションの指定

標準的な手法では、ソース制御システムでの資産のチェックインおよびチェックアウトのプロセス中にコメントを指定します。

4.19　「Workspace」ビューの最新表示

複数のユーザーが資産をチェックインおよびチェックアウトしていると、あるユーザーのローカル・ワークスペースで、他のユーザーによってチェックインされた資産が欠落する場合があります。

このセクションでは、ClearCase リポジトリーの最新状態を反映してローカルのソース管理ワークスペースを更新する手順を説明します。この手順は、図 4-58 (119 ページ) および図 4-59 (120 ページ) に示されています。

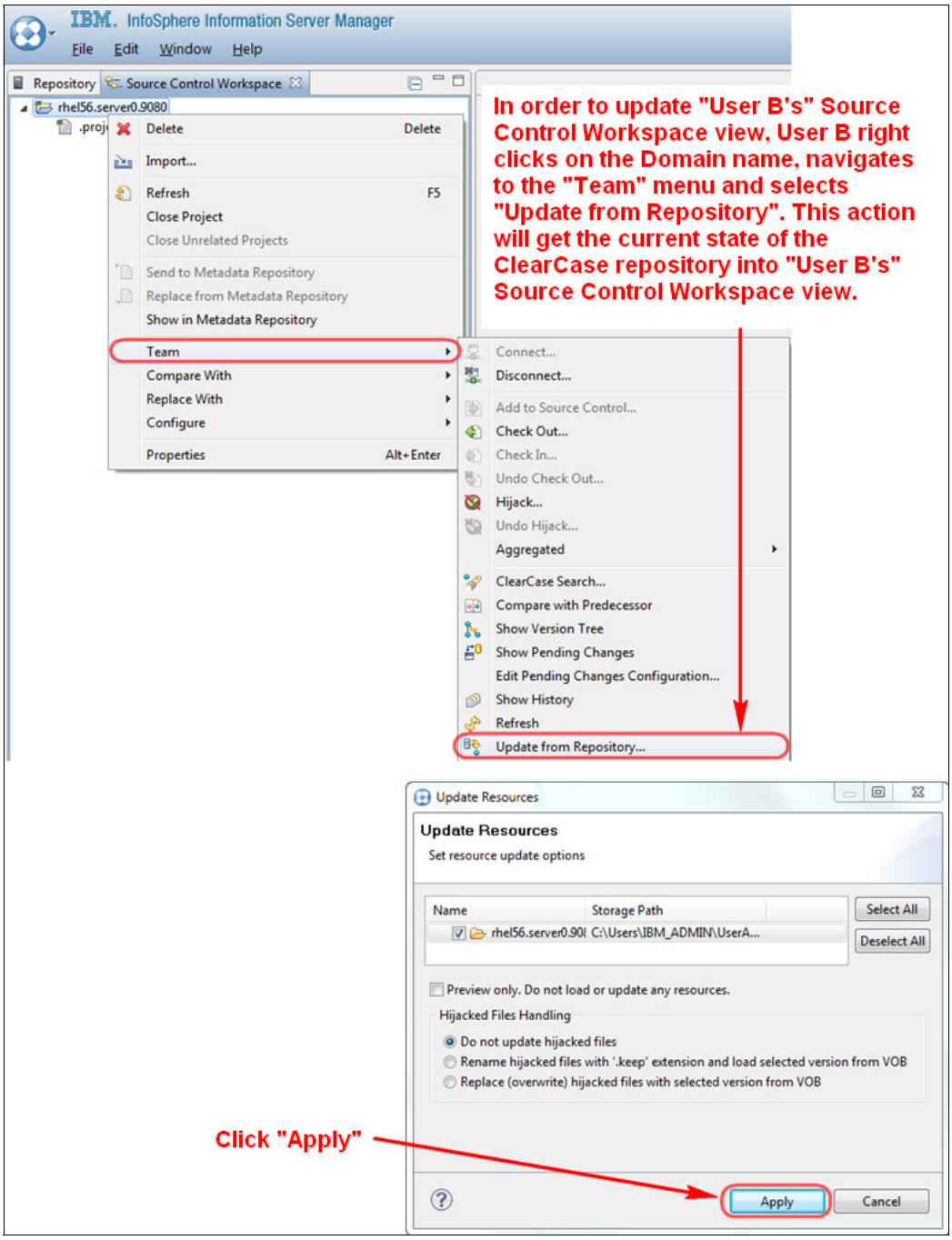


図 4-58　「Workspace」の最新表示 (ステップ 1/2): 「Update from Repository」

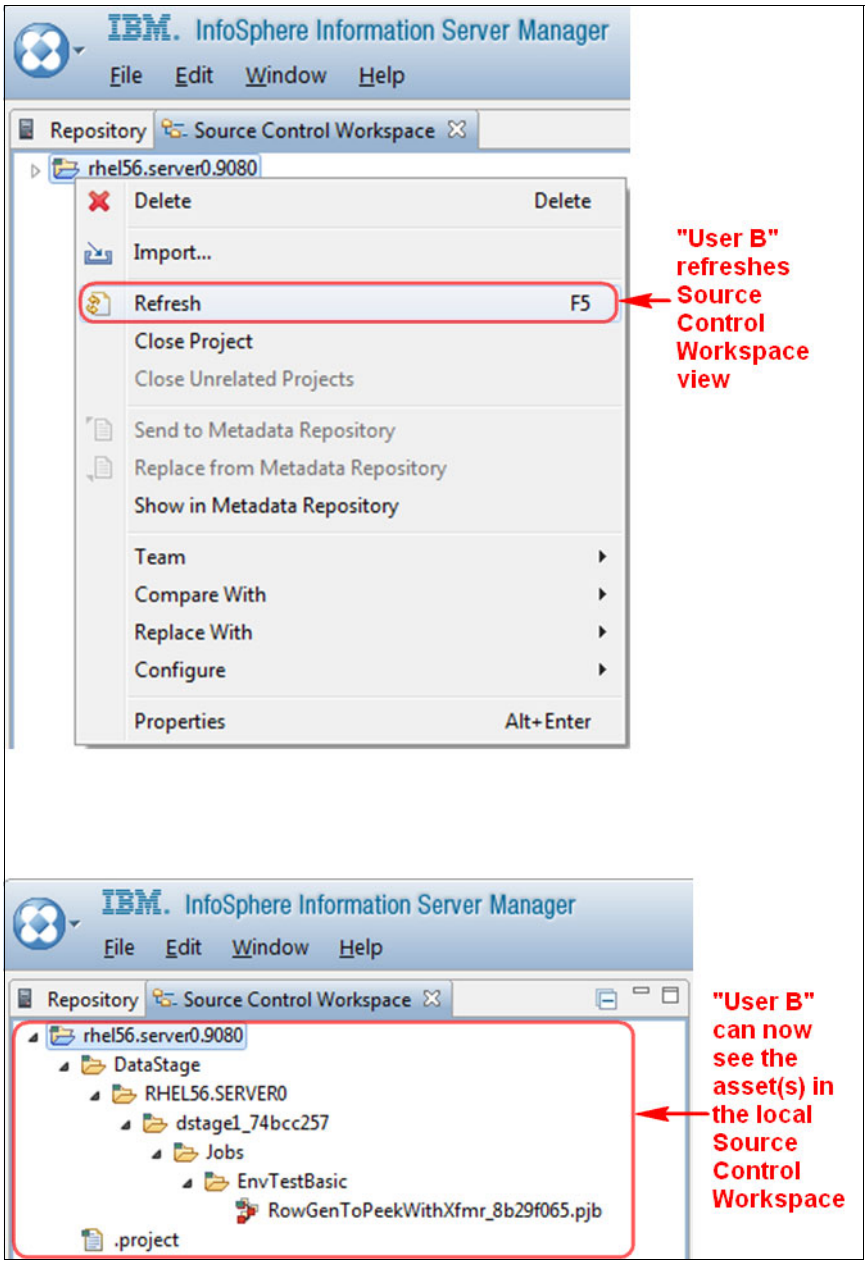


図 4-59　「Workspace」の最新表示 (ステップ 2/2): 「Refresh」オプションの選択

4.20 デプロイメント・パッケージの作成とビルド

このセクションでは、パッケージを作成およびビルドするステップを説明します。これらのステップは、図 4-60 (121 ページ) から図 4-62 (123 ページ) に示されています。パッケージを作成した結果、定義が生成されます。ビルド・オプションは実際にパッケージをビルドします。このパッケージは、その後、デプロイメントに使用できます。

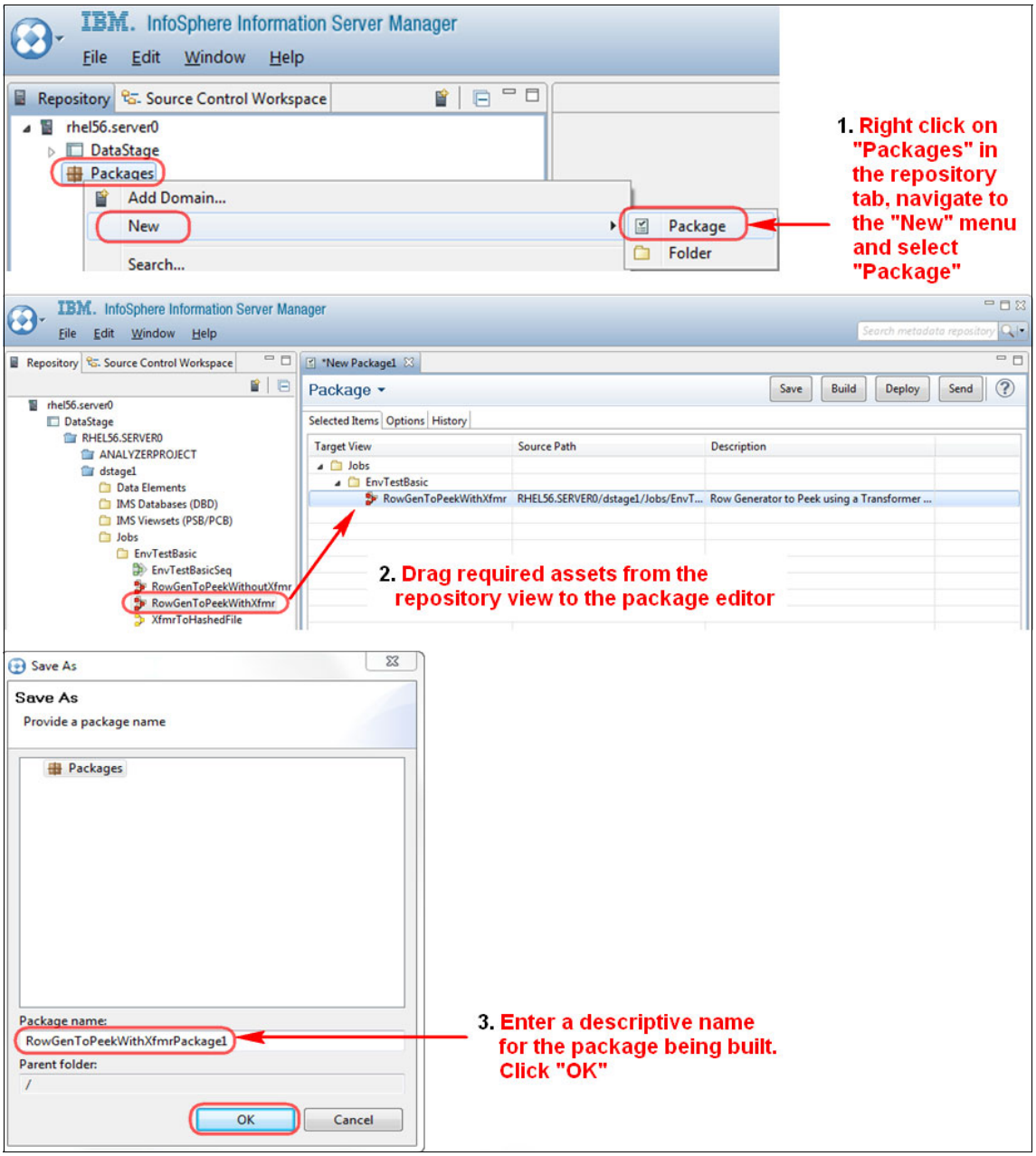


図 4-60　パッケージの定義と保存

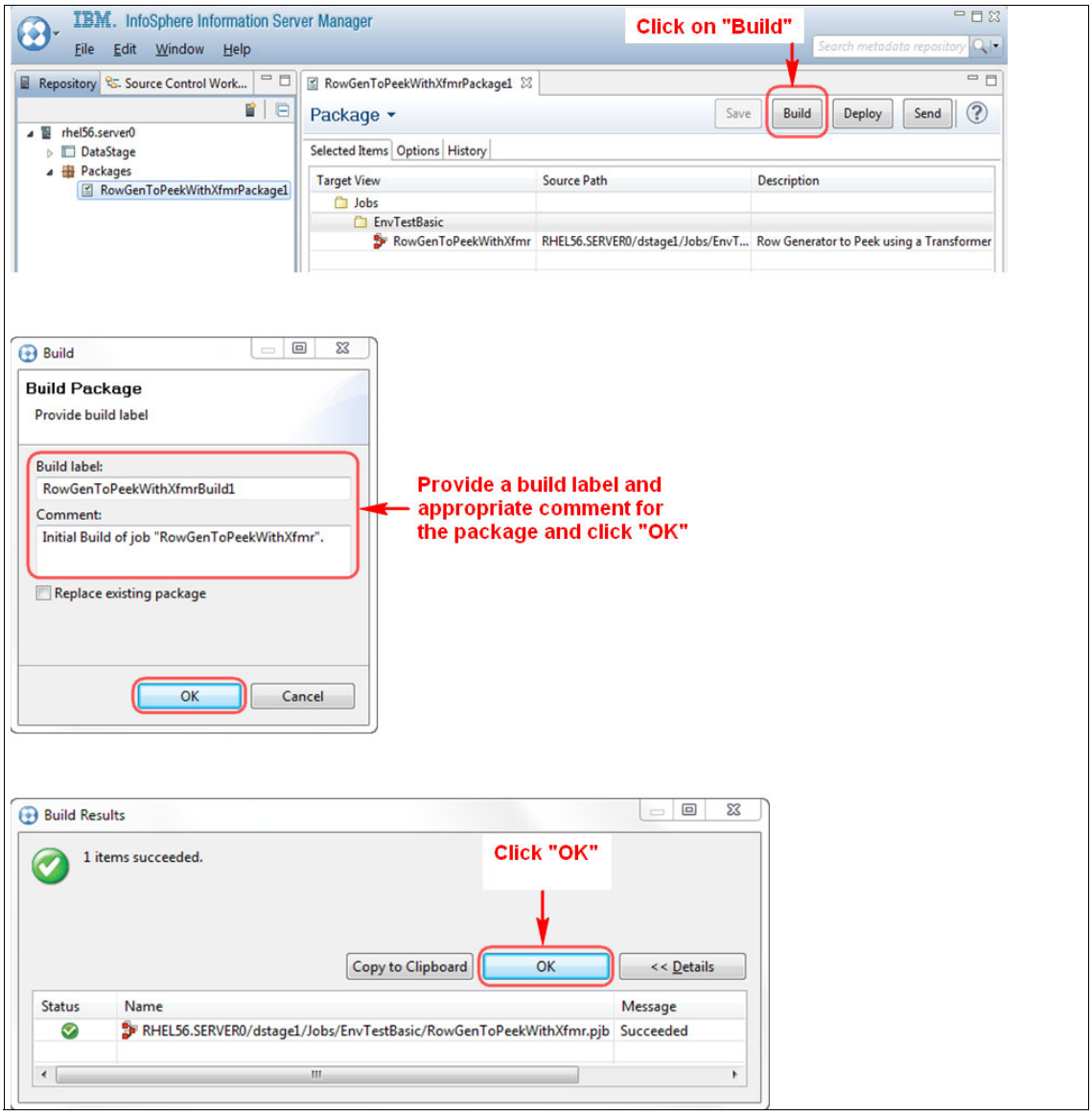


図 4-61　パッケージのビルド

レプリカ：パッケージがビルドされる際、パッケージのコンテンツ (資産) は、ビルド時における Information Server リポジトリー内の資産のバージョンのレプリカです。これらは、パッケージ定義が作成された時点の資産のバージョンではありません。混同しないように注意してください。

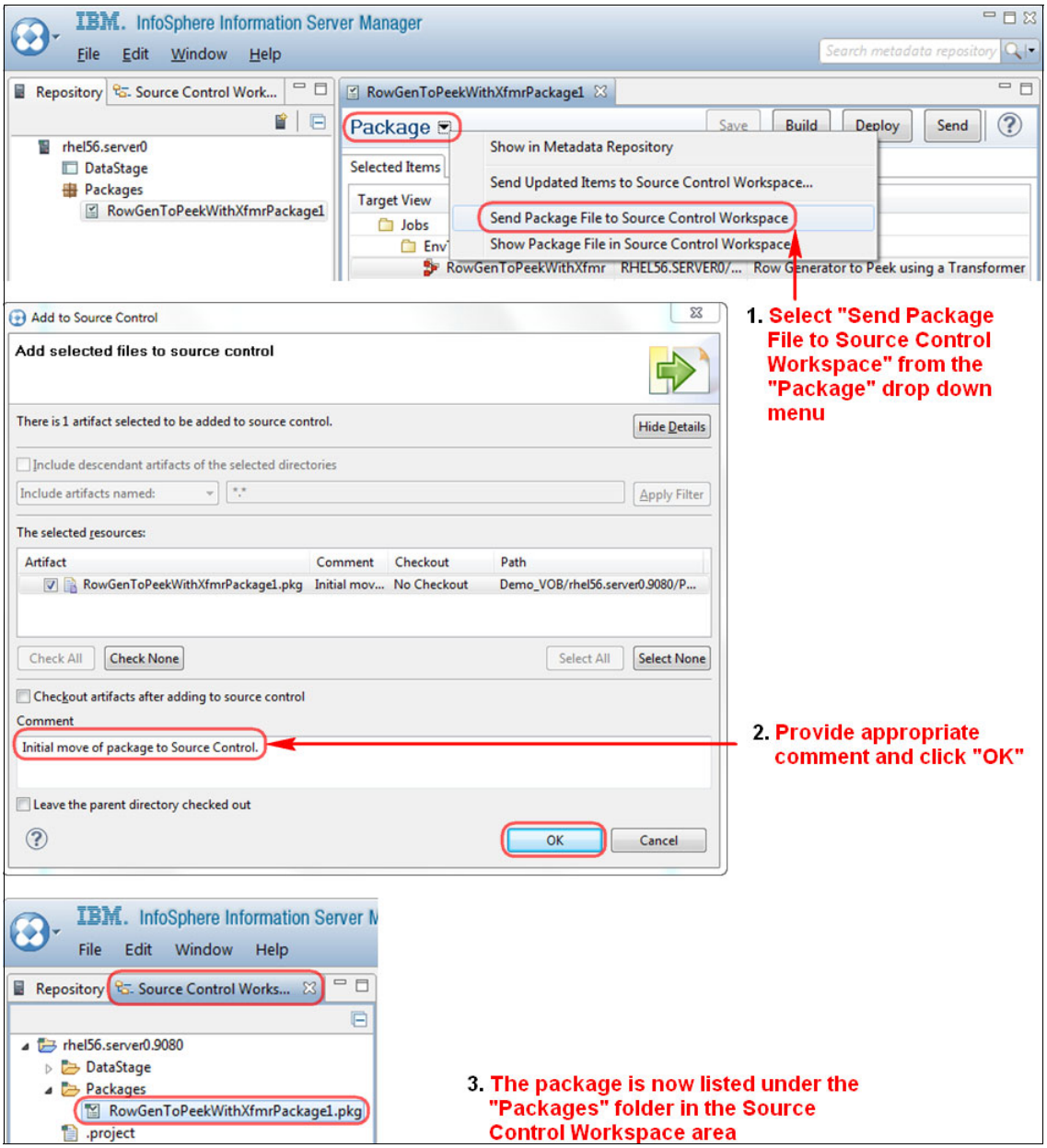


図 4-62　「Source Control Workspace」へのパッケージの送信

4.21　「Deploy」および「Send」オプションの使用

図 4-63 に示されている「Deploy」ボタンは、同じ IS ドメイン内にあるプロジェクトのみに対して機能します。

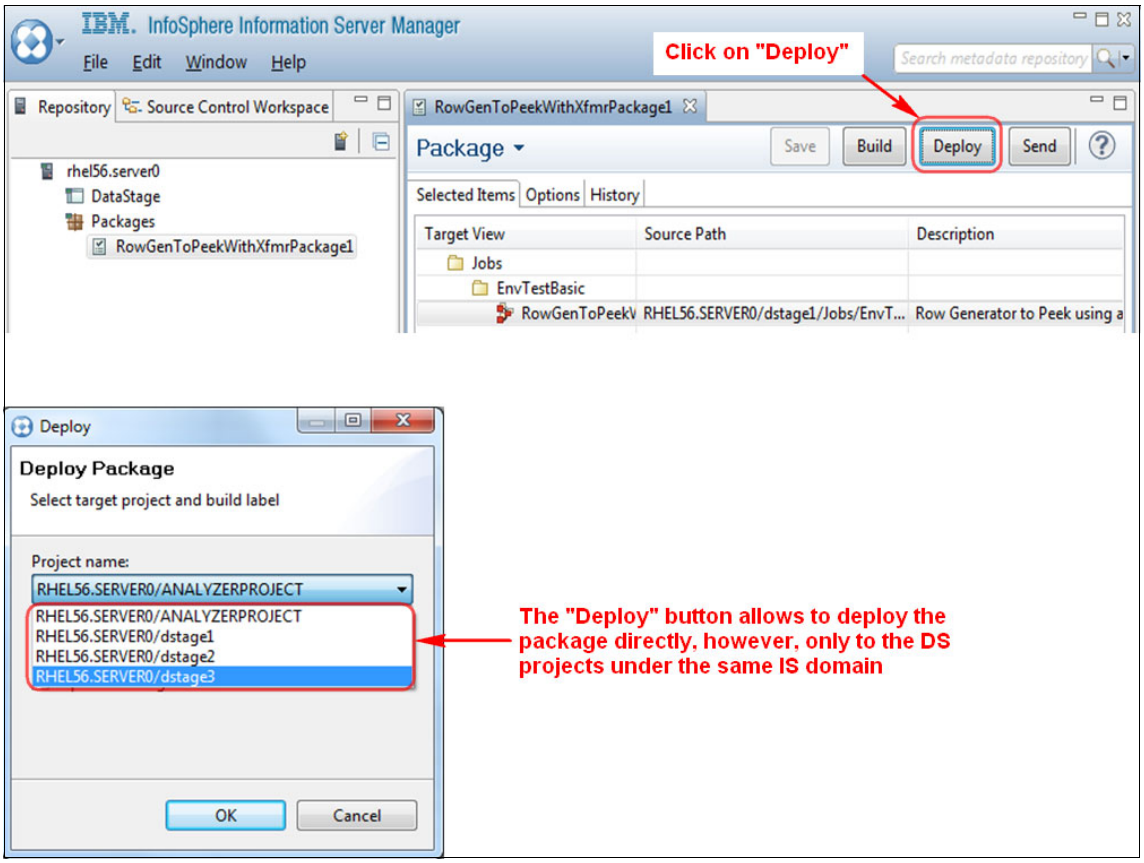


図 4-63　「Deploy」ボタンの使用

パッケージを別のドメインにデプロイするには、「**Send**」をクリックします。これにより、パッケージがローカル Windows ファイル・システムに保存されます。その後、プロモーション・パス (テスト、QA、本番) の他の環境のリリース管理者がアクセスできる共通フォルダーに (コピーまたは FTP のいずれかの操作によって) このフォルダーを配置できます。

後続のリリース管理者は、「**Deploy from file**」オプションを使用します。このオプションについては、4.22『ClearCase Explorer によるデプロイメント・パッケージのプロモート』（125 ページ）(図 4-67 (129 ページ) および図 4-68 (130 ページ)) で説明しています。

あるいは、プロモーション・プロセスで ClearCase を使用できます (4.22『ClearCase Explorer によるデプロイメント・パッケージのプロモート』（125 ページ）を参照)。この方法では、デプロイメント・パッケージは、FTP によってコピーされず、ClearCase のチェックイン機能とチェックアウト機能を除く他のどの方法でも交換されません。

「Send」ボタンの使用は、図 4-64 に示されています。

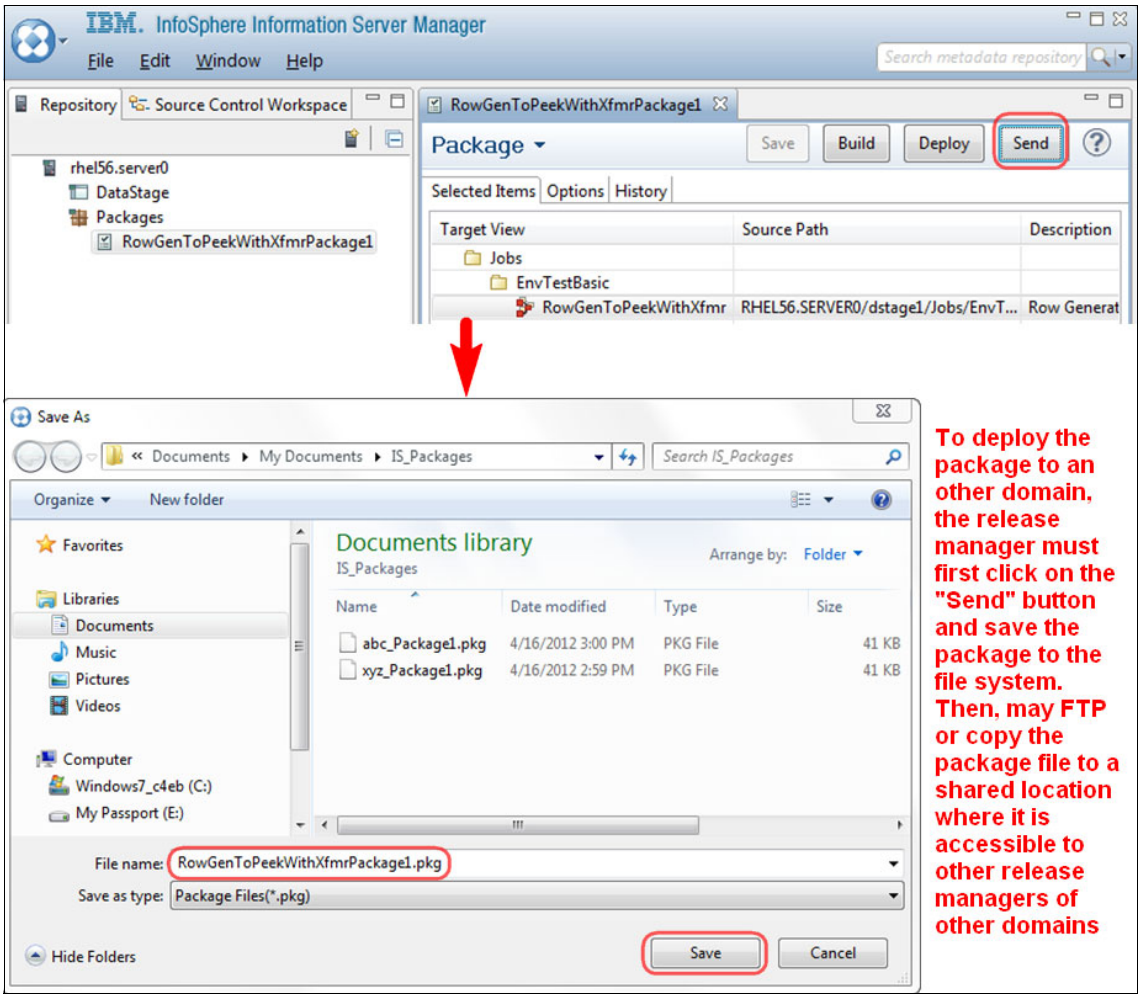


図 4-64　「Send」ボタンの使用

「Send」ボタンを使用する代わりに、4.21『「Deploy」および「Send」オプションの使用』（123 ページ）のアプローチを使用できます。この方法では、(複数のリリース管理者が関与する) プロモーション・プロセス全体を通してソース制御のアプローチに従い、ClearCase のチェックインおよびチェックアウトのプロセスに完全に依存します。

4.22 ClearCase Explorer によるデプロイメント・パッケージのプロモート

このセクションで概要を示すアプローチでは、パッケージのプロモーション・プロセスで ClearCase に完全に依存します。

1. 開発のリリース管理者は、まだ、前述の方法 (図 4-62 (123 ページ)) でパッケージを作成して「Source Control Workspace」に送信しています。この方法では、パッケージは ClearCase にアップロードされます。
2. テストと本番のリリース管理者は、開発環境の ClearCase のパッケージ・フォルダーに対するアクセス権を付与される必要があります。このようにして、これらの管理者は ClearCase からデプロイメント・パッケージをチェックアウトできます。
3. テストと本番のリリース管理者は、IS Manager の GUI (「Repository」ビュー) を使用してパッケージをチェックアウトするのではなく、ClearCase パースペクティブで ClearCase Navigator を使用してデプロイメント・パッケージをチェックアウトします。このステップについては、4.22.1

『ClearCase Explorer によるデプロイメント・パッケージのチェックアウト』（127 ページ）で説明しています。これらの管理者は、ClearCase で開発フォルダー構造のビューを作成できる必要があります。

4. チェックアウト時に、デプロイメント・パッケージはローカル・ワークスペース・エリアに保存されます。
5. これらのリリース管理者は、「**Deploy from file**」オプション (4.22.2『チェックアウトされたパッケージのデプロイ』（128 ページ）を参照) を使用して、ローカル・ファイル・システムをナビゲートしてデプロイメント・パッケージを見つけ、それぞれのシステム (テストまたは本番) にデプロイすることができます。
6. デプロイメント・パッケージは ClearCase にアップロードされるため、リポジトリーの ClearCase ラベルを ClearCase リポジトリー内の資産に直接適用できます。これらのラベルは、開発からテスト、そして本番までのすべてのプロモーション・プロセスの間、各デプロイメント・パッケージの状況の指標として使用できます。

4.22.1 ClearCase Explorer によるデプロイメント・パッケージのチェックアウト

このセクションでは、IS Manager の ClearCase Explorer パースペクティブを使用して開発環境からパッケージを取得するプロセスを説明します。図 4-65 および図 4-66 (128 ページ) を参照してください。

テストと本番のリリース管理者が、ClearCase で開発環境に関連するデプロイメント・パッケージ資産のビューを作成することを許可されていることが前提となります。つまり、これらの担当者は、ClearCase で直接的にデプロイメント・パッケージをチェックインおよびチェックアウトできます (そもそも開発用の IS 環境の資格情報がないため、IS Manager の「Repository」または「Workspace」ビューは使用しません)。

ClearCase で開発環境のその他のフォルダー (ジョブやテーブル定義などに関連するもの) へのアクセス権が付与されてはなりません。

ここで言う許可とは、ClearCase に関連するものです。繰り返しになりますが、テストおよび本番の管理者は、開発用の IS インスタンスへのアクセス権をまったく持っていません。唯一のアクセス権は、ClearCase ツール (ClearCase Explorer パースペクティブなど) を使用してデプロイメント・パッケージをチェックインおよびチェックアウトできることです。

図 4-65　ClearCase Explorer パースペクティブのオープン

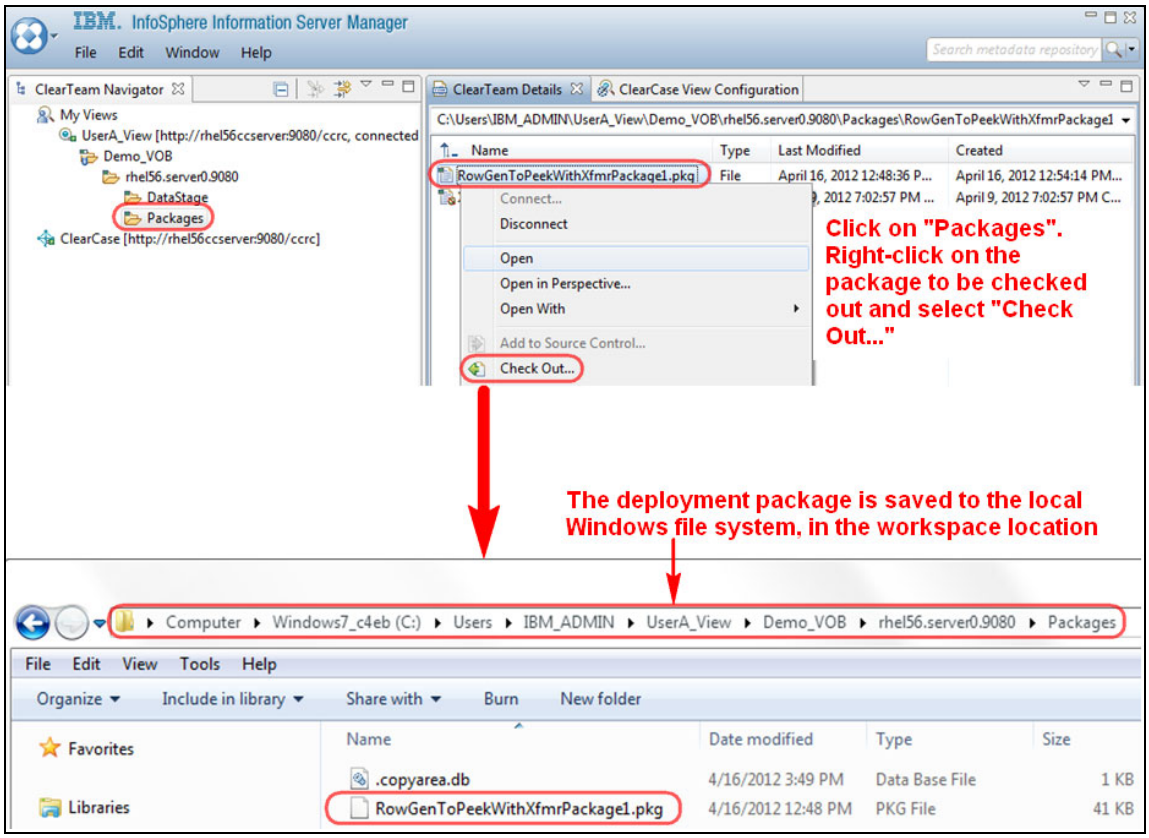


図 4-66　ClearCase Explorer パースペクティブによるパッケージのチェックアウト

図 4-66 に示されているビューは、開発環境を想定しています。この図のステップを実行するユーザーは、開発からパッケージを見つけるテストまたは本番のリリース管理者です (繰り返しになりますが、ClearCase リポジトリーから直接的に取得します)。この担当者は、パッケージのみのチェックインとチェックアウトを許可されています (開発のその他の資産に対しては許可されていません)。

ClearCase Navigator で、ユーザーは、ビューをナビゲートしてパッケージ・フォルダーを見つけ、パッケージ名を右クリックして、「**Check Out**」を選択します。このステップにより、パッケージは、テストまたは本番のリリース管理者が作業しているローカル Windows ファイル・システムに保存されます。

次に、テストまたは本番のリリース管理者は、4.22.2『チェックアウトされたパッケージのデプロイ』（128 ページ）で説明されているプロセスを使用して、ローカル側に保存されたパッケージ・ファイルをデプロイすることができます。

4.22.2 チェックアウトされたパッケージのデプロイ

このセクションでは、ローカル側に保存されたパッケージをデプロイするステップを説明します。このセクションは、4.22.1『ClearCase Explorer によるデプロイメント・パッケージのチェックアウト』（127 ページ）から始まる、パッケージが ClearCase からチェックアウトされてローカル側に保存されるプロセスの続きです。

図 4-67 (129 ページ) では、テストまたは本番のリリース管理者は既にターゲットのテスト環境または本番環境に接続されています。このプロセスでは以下のステップを実行します。

1. 特定のターゲット・ドメインの「Repository」ビューで「**Packages**」を右クリックします。
2. 「**Deploy Package From**」→「**File**」を選択します。

3. フォルダーおよびファイル名を選択します。
4. 「**Open**」をクリックします。

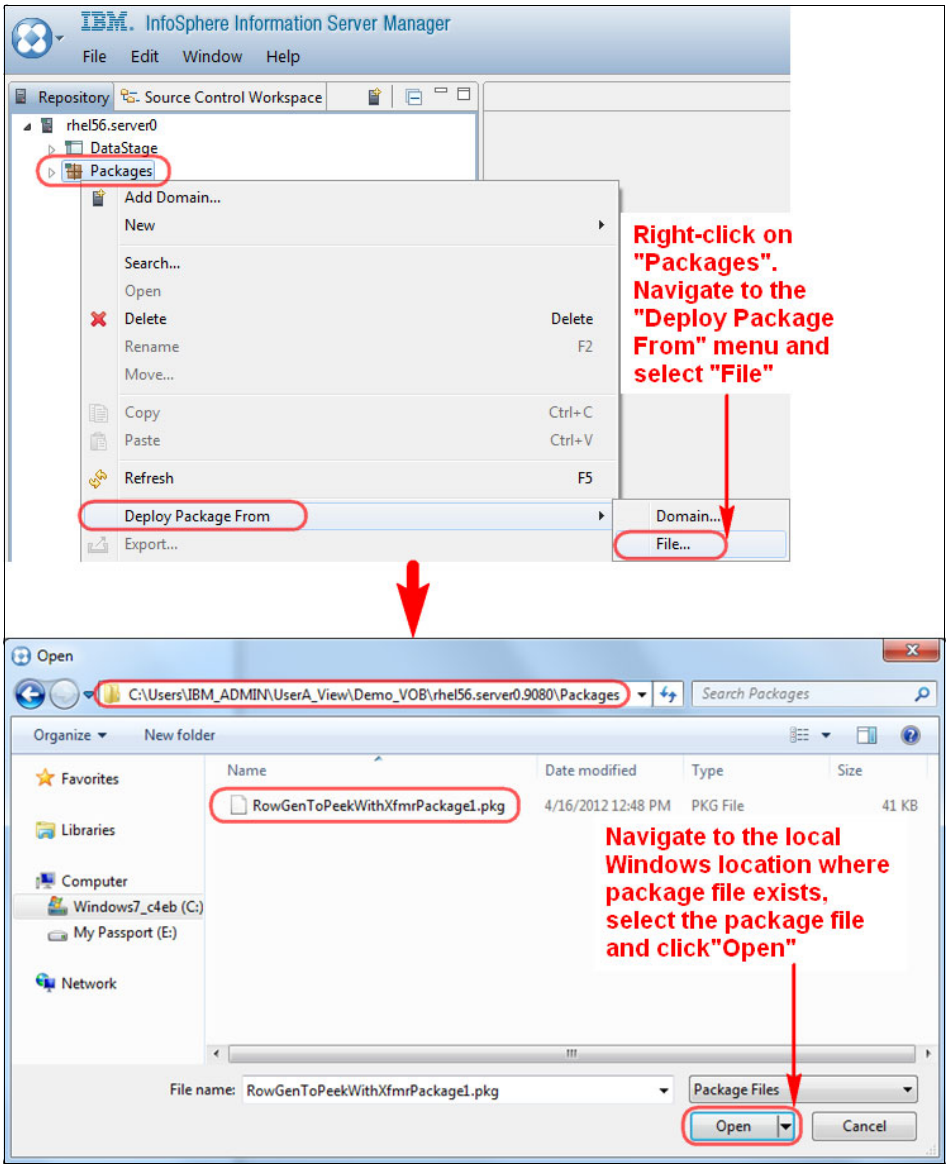


図 4-67　ClearCase Explorer でのチェックアウトされたパッケージのデプロイ : パッケージ・ファイルの選択

この続きは、図 4-68 (130 ページ) に示されています。パッケージが Information Server Manager のターゲット・ドメインの「Repository」ビューにアップロードされた後、「**Deploy**」をクリックします。ポップアップ・ウィンドウで、ターゲットの DataStage プロジェクトを選択して、「**OK**」をクリックします。すると、資産は、その DataStage プロジェクトにインポートされます。

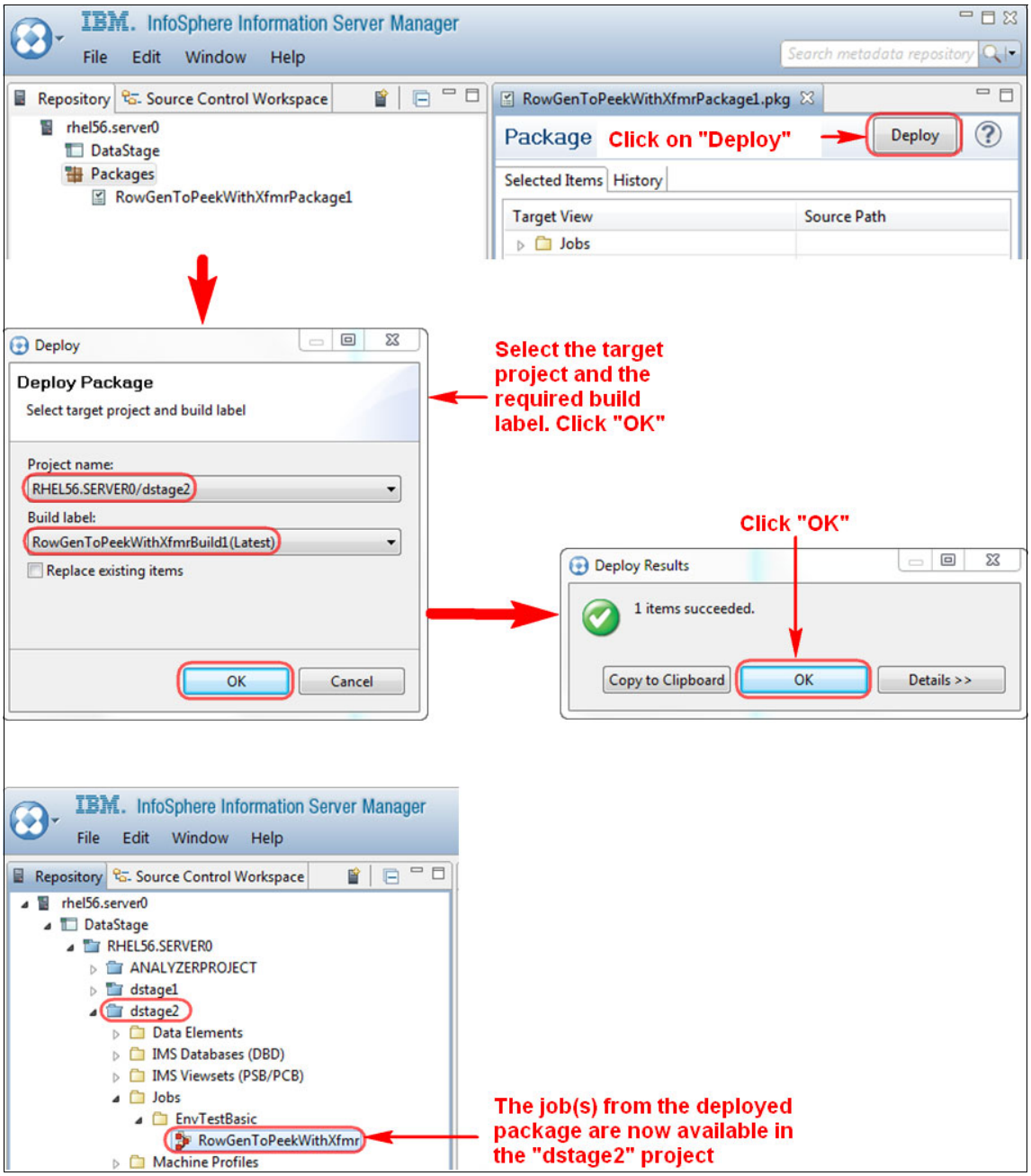


図 4-68　ClearCase Explorer でのチェックアウトされたパッケージのデプロイ : デプロイメントの実行

4.23　テスト・データのデプロイメント

本書の前のセクションで、情報統合プロジェクトのライフサイクルを通じて使用されるいくつかの Information Server 環境の要件について説明しました。例えば、4 つの異なる Information Server インストール済み環境は、開発、テスト、疑似本番、および本番に対応しています。1 つのインストール済み環境の中で DataStage、QualityStage、および Information Analyzer の複数のプロジェクトが存在できます。

それぞれの環境について、組織は、その環境で DataStage プロジェクト (複数可) によって使用される関連テスト・データを格納する物理的なソースとターゲットのデータベースのプロビジョニングを適宜に検討する必要があります。一般に、本番のソースとターゲットのデータベースが認識されている統合プロジェクトの段階では、よく使用されるアプローチは、関連するソースとターゲットのデータベースのポイント・イン・タイム・コピーをプロビジョンする

ことです。これらのコピーは、小規模の環境のテスト要件に十分であることが分かっているためです。テスト・シナリオでより最近のデータが必要になった場合、コピーを時々リフレッシュできます。

このタイプのテスト・データのプロビジョニングでは、以下の考慮事項があります。

- サイジングと関連性

  ソースとターゲットのデータベースは、一般に Information Server とは別のインフラストラクチャーでホストされる必要があり、多くの場合は環境ごとに分離されています。

  そのため、本番データベースの完全なコピーを作成する物理ホスト・システムのサイジング要件と、サブセットでテストのニーズに対応できる場合のトレードオフを検討してください。サブセットのメリットは、物理サイジングの影響を抑えられることです。デメリットは、サブセットの選択方法を設計しなければならない可能性があることです。

  サブセットの使用を目的としたデータの選択方式の 2 つの例は、指定の存続期間よりも前のタイム・スタンプを持つデータを選択するか、本番データベースのすべての表の関連サブセットをコピーすることです。

  データを削減するどの方法でも、重要な考慮事項は、サブセットで関連シナリオが除外されないようにすることです。ただし、このアプローチは、情報統合プロジェクトのライフサイクルにおいて、まだデータを理解しようとしている初期段階で懸念事項となる可能性があります。このアプローチでは、開発サイクルの後の時点まで関連性が理解されないデータが除外される可能性があり、その結果としてプロジェクト・コストに影響します。

  ソース・データとルールが十分に理解されていない場合 (一般的な状況)、Discovery などのツールを使用してプロファイリングを実施することが重要です。

  疑似本番 環境では、制御されたパフォーマンス・テストで本番のパフォーマンスを再現しようとする傾向があるため、ソースとターゲットのデータベースの完全なボリューム・コピーが不可避となる可能性があります。

- 機密データの問題

  お客様の本番環境では通常、さまざまなタイプの機密データが保持されています。例えば、商業上の機密データには、顧客データ保護の問題、クレジット・カード番号、またはデータ・プライバシーに関する政府の基準への準拠性などのその他の特別なセキュリティー上の考慮事項があるデータが含まれます。

  本番データベースのプロビジョニング・コピーが必要となる本番以外の環境では、データ・マスキング技法を使用して、関連する問題の管理に役立てることができます。*データ・マスキング* は、データ・ストア内の特定のデータ・エレメントを覆い隠す技法です。

  物理的なマスキング・ソリューションのアクセスと場所を検討します。これには、マスキングを必要とする機密データへのアクセス権が必要です。

  各技法の特性としてプロセスが反復可能でなければならないという点に注意してください。テスト・データのリフレッシュなどを行うときに再適用する必要が生じる可能性があるためです。

  マスキング・ソリューションの選択または設計においては、以下の問題が生じます。

  – リレーショナル・キーの一部であるデータは、親子関係の必要性から、一貫した方法で変更される必要があります。

  – 特定のパターン、フィールド長、サブストリング、タイプの頻度には、後続の DataStage ジョブ、開発される品質ルール、またはデータ・プロファイリング・アクティビティーの設計に関連する組み込みデータが含まれている可能性があります。

InfoSphere 環境でのデータ・マスキングのアプローチには、以下の技法を使用できます。

– DataStage ジョブのカスタム開発

  DataStage の標準機能を使用して、必要なルールを DataStage ジョブでコーディングできます。前述の問題や特定の要件の複雑さを考慮すると、DataStage の開発の効率性にかかわらず、この作業自体に多大な労力が必要となる可能性があります。(任意の新規または一括テスト・データを生成する際、行生成プログラムの DataStage も使用されます。)

– IBM InfoSphere DataStage Pack for Data Masking

  このパックは、DataStage のアドオン・パックであり、データ・マスキング・ポリシーの適用によって企業データのマスキングを可能にします。このパックは、DataStage デザイナー・キャンバスおよび関連するマスキング・ポリシー・エディターで追加のステージを提供します。

  カスタム・ポリシーの定義に加えて、以下のマスキング・ポリシーを使用できます。

  • クレジット・カード番号データ・マスキング・ポリシー

    このポリシーは、ソース・データに基づいてクレジット・カード番号に適切なマスクを生成します。IBM InfoSphere DataStage Pack for Data Masking は、例えば、American Express、MasterCard、Visa、および Discover のクレジット・カードのデータ・マスキングをサポートします。

  • E メール・アドレス・データ・マスキング・ポリシー

    このポリシーは、ソースの E メール・アドレスに適切なマスクを生成します。E メール・アドレス全体、ユーザー名のみ、またはドメイン・ネームのみをマスキングできます。

  • 米国国民 ID データ・マスキング・ポリシー : 国民 ID (米国)

    米国の国民 ID 番号は社会保障番号です。このポリシーは、ソース・データに基づいて社会保障番号に適切なマスクを生成します。

  • カナダ国民 ID データ・マスキング・ポリシー : 国民 ID (CA)

    カナダの国民 ID 番号は社会保険番号です。このポリシーは、ソース・データに基づいて有効なカナダ社会保険番号を生成します。

  • フランス国民 ID データ・マスキング・ポリシー : 国民 ID (FR)

    フランスの国民 ID 番号は、フランス国立統計経済研究所の番号です。このポリシーは、ソース・データに基づいて有効なフランス国立統計経済研究所の番号を生成します。

  • イタリア国民 ID データ・マスキング・ポリシー : 国民 ID (IT)

    イタリアの国民 ID 番号は財務コードです。このポリシーは、ソース・データに基づいて有効な財務コード番号を生成します。財務コード番号がマスキングされると、氏名を含む部分はソースからコピーされ、その他の部分はマスキングされます。同じソース・データに対して何回実行しても結果は常に同じです。

– スペイン国民 ID データ・マスキング・ポリシー : 国民 ID (ES)

  スペインの国民 ID 番号は、財務 ID 番号 (NIF) または外国人 ID 番号 (NIE) です。NIF はスペイン国民に与えられ、NIE は外国人居住者に与えられます。

– 英国国民 ID データ・マスキング・ポリシー : 国民 ID (UK)

  英国の国民 ID 番号は、国民保険番号 (NINO) です。このポリシーは、ソース・データに基づいて有効な国民保険番号を生成します。

– 日付 / 存続期間データ・マスキング・ポリシー

このポリシーは、ソース・データ値に基づいて新しい日付を生成します。日付 / 存続期間データ・マスキング・ポリシーは、ランダムな日付を生成しません。

– 反復可能置換データ・マスキング・ポリシー

  このポリシーは、ソース・データの形式でソース・データをマスキングします。反復可能置換データ・マスキング・ポリシーには、必ず入力データを指定する必要があります。このデータ・マスキング・ポリシーを使用して、1 次キーまたは外部キーなどのキーを変換できます。

– ランダム置換データ・マスキング・ポリシー

  このポリシーは、ソース・データの別の実行のために別の形式でソース・データをマスキングします。

– ハッシュ・ルックアップ・データ・マスキング・ポリシー

  このポリシーは、データベースの参照表を使用して入力列をマスキングします。ソース列の値に基づいてハッシュ値を計算し、ハッシュ・キー列がハッシュ値と一致するレコードを取得します。

– その他のマスキングの可能性として、マスキングされた 10 進変換、マスキングされた文字、およびマスキングされた左右変換などが挙げられますが、これらに限られません。

  図 4-69 (133 ページ) に、シンプルな DataStage ジョブを示します。ここでは、Data Masking Pack が、Data Masking ステージを使用する Information Server のインストールの一部としてインストールされています。このジョブは、入力データを読み取り、その出力を順次ファイルに書き込みますが、関連するステージを使用することにより、サポートされているソースとターゲットのデータベースでの読み取りと書き込みと同様に簡単に実行できます。図 4-69 (133 ページ) に示すように、Data Masking ステージではオプションでリジェクトリンクを指定できます。その場合はソース・レコードが予期されるパターンと一致しておらず、さらなる調査のために行をリジェクトできます。

  データ・マスキングを行う DataStage ジョブは、DataStage によってサポートされている幅広い種類、タイプ、バージョンのソースとターゲットのデータベースへの接続をサポートします。

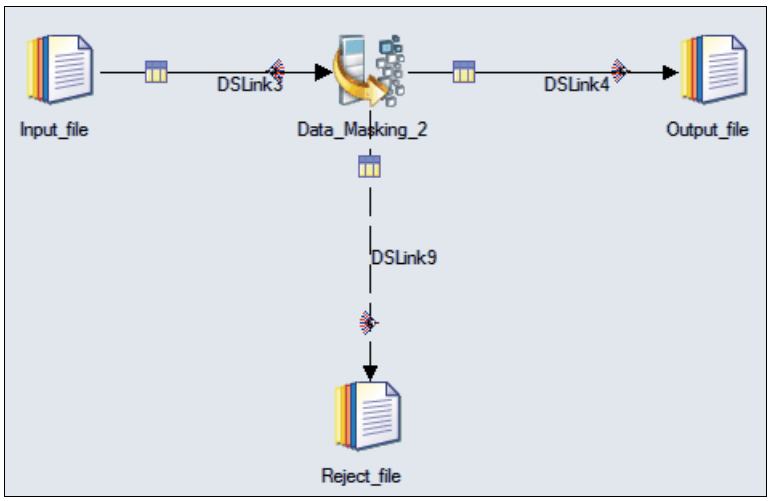


図 4-69　DataStage のデータ・マスキング・ステージ

  データ・マスキング・ステージの中で、データ・マスキング・ポリシー・エディターを使用して、必要に応じて各列にマスキング・ルールを別々に設定できます。

– IBM InfoSphere Optim Data Masking Solution の使用

Optim は、InfoSphere 製品で使用できる追加のスイートであり、特にデータ・マスキングを目的としています。ここでは、データ・マスキングに対するエンタープライズ・ソリューションを背景として簡単に説明します。

InfoSphere Optim Data Masking Solution を使用すると、ユーザーは、さまざまな実績あるデータ変換技法を適用して、実際の機密データを、概念的には正確かつ現実的な架空データに置き換えることができます。ユーザーは、単一データベースまたは複数の関連するシステムでデータをマスキングできます。InfoSphere Optim のマスキング技法のシンプルな例として、サブストリング、演算式、乱数または連番の生成、日付の経年処理、および連結が挙げられます。コンテキストを認識するマスキング機能により、マスキングされたデータが元の情報の外観を維持することが保証されます。このような機能により、誕生日、銀行の口座番号、所在地住所と郵便番号の組み合わせ、および国民 ID といった多くのタイプの機密情報を容易に非特定化できます。

InfoSphere Optim は、お客様の現在、そして将来の要件に簡単に適合できる柔軟な機能を備えた、スケーラブルなデータ・マスキング・ソリューションを提供します。InfoSphere Optim は、IBM DB2、Oracle、Sybase、Microsoft SQL Server、IBM Informix®、IBM IMS、IBM Virtual Storage Access Method (VSAM)、Teradata、Adabas、Microsoft Windows、UNIX、Linux、および IBM z/OS® を含む、すべての主要なエンタープライズ・データベースとオペレーティング・システムをサポートします。すべてのカスタム・アプリケーションとパッケージ・アプリケーションに対するデータ管理サポートを提供するほか、InfoSphere Optim には、今日使用されている大半の最新のエンタープライズ・リソース・プランニング (ERP) およびカスタマー・リレーションシップ・マネジメント (CRM) アプリケーションをサポートするためのメタモデルの知識が組み込まれています。例として、SAP、Oracle E Business Suite、PeopleSoft Enterprise、JD Edwards EnterpriseOne、Siebel、および Amdocs CRM が挙げられます。また、HIPAA、GLBA、DDP、PIPEDA、Safe Harbour、PCI DSS などのプライバシーの規制への準拠もサポートします。

4.24 その他の資産のライフサイクル管理

ここまでのセクションでは、DataStage および QualityStage の資産のライフサイクル管理について説明しました。

Information Server は、DataStage および QualityStage 以外の資産も含まれる情報統合ランドスケープで発生する多くの資産の管理を支援します。

表 4-2 に、Information Server が管理できる幅広い資産の種類をカテゴリー別にグループ化してリストします。

表 4-2 Information Server の資産のタイプ

資産カテゴリー	資産
Business Glossary	► カテゴリー ► 用語
Fastrack	► マッピング・コンポーネント ► マッピング構成 ► マッピング仕様 ► プロジェクト・テンプレート ► プロジェクト ► プロジェクトに関連付けられているロールの割り当て、レポート、共通メタデータ、グロッサリー資産、および IBM InfoSphere DataStage と QualityStage の資産

資産カテゴリー	資産
Information Analyzer	► プロジェクト (分析結果、データ・ルール、およびデータ・クラスなど、プロジェクトに関連付けられている資産を含む) ► すべてのデータ・クラス (関連付けられているプロジェクトに関係ありません) ► プロジェクトに関連付けられているレポートおよび共通メタデータ
共通メタデータ資産	► ビジネス・インテリジェンス (BI) 資産 : BI モデル BI コレクション キューブ BI レポート BI レポート照会 ► 実装されているデータ・リソース : ホスト・コンピューター データベース データベース・スキーマ ストアード・プロシージャー データベース表 データ・ファイル データ・ファイル構造 データ接続 データ・アイテム定義 ► 論理データ・モデル資産 : 論理データ・モデル 論理エンティティー 論理関係 エンティティー一般化階層 論理領域 サブジェクト・エリア ダイアグラム ► 物理データ・モデル資産 : 物理データ・モデル 設計表 設計ストアード・プロシージャー 物理領域 ► 各種の資産 データ接続 カスタム属性 連絡先ライブラリー
レポーティング資産	► レポート ► レポート結果
セキュリティー資産	► ユーザー (ロール、資格情報、資格情報マッピングの有無は関係ありません) ► グループ (ロールの有無は関係ありません)

資産カテゴリー	資産
DataStage および QualityStage の資産 (説明については前のセクションを参照)	カスタム・フォルダー (プロジェクト・ディレクトリー内の外部ファイル) ► データ接続 ► データ・エレメント ► データ品質仕様 (ルックアップ・テーブル、ルール・セット、およびマッチング仕様) ► IMS データベース ► IMS ビュー・セット ► ジョブ (メインフレーム、パラレル、シーケンス、およびサーバー) ► ジョブ実行可能ファイル ► マシン・プロファイル ► パラメーター・セット ► ルーチン (メインフレーム、パラレル、およびサーバー) ► 共有コンテナー (パラレルおよびサーバー) ► ステージ・タイプ ► 表定義 ► 変換 ► 表定義に関連付けられている共有表および関連する共通メタデータ資産

134 ページの表 4-2 を確認して、特に DataStage および QualityStage の資産のライフサイクルとメタデータ関連資産のライフサイクルと比較すると、認識すべき重要な概念は、メタデータのライフサイクルが DataStage および QualityStage よりもはるかに長いことです。

以下の例を検討してください。

- 一部の Business Glossary 用語は、所有組織が存続する限り適用可能です。
- ソースとターゲットのデータベース表のテクニカル・メタデータは、そこから抽出する DataStage プロジェクトよりも前に定義され、存在している必要があります。実際、DataStage プロジェクトのライフサイクルを通じて、DataStage ジョブの多くのバージョンがその同じ表からの読み取りに関与します。
- Information Analyzer のデータ品質ルールは、DataStage で通常の ETL ソリューションを開発するニーズから分離して存在できます。

137 ページの第 5 章、『メタデータのデプロイメント・アーキテクチャー』では、134 ページの表 4-2 で説明した幅広い情報統合資産を考慮に入れた Information Server のデプロイメント・トポロジーの選択肢について説明します。

5

メタデータのデプロイメント・アーキテクチャー

すべての InfoSphere Information Server 製品モジュールはメタデータを生成し、使用します。生成されるメタデータは、Information Server リポジトリーで保持されます。この Information Server 共有リポジトリーを介して、Information Server は、製品モジュールとユーザー (アナリスト、データ・スチュワード、対象分野の専門家、および開発者など) の間のコラボレーションを実現します。このふるまいは、開発環境、テスト環境、本番環境、および企業にあるその他の環境など、すべての Information Server インスタンスに当てはまります。ただし、開発ライフサイクルは、Information Server の各資産によって異なります。例えば、DataStage ジョブのデザイン (コード) とデータ・ソースのテクニカル・メタデータとでは異なります。

Information Server 資産のさまざまなライフサイクルは、Information Server 製品モジュールのデプロイメント・アーキテクチャーに影響を与える可能性があります。ビジネス要件に応じて、Information Server 製品モジュールは、環境の一部または全体にデプロイされます。使用するフィーチャーと活用するメタデータのコラボレーションのレベルに応じて、同じ製品モジュールでも異なる方法でデプロイされる場合があります。したがって、標準的な開発環境 (DEV)、テスト環境 (TEST)、および本番環境 (PROD) のデプロイメント・アーキテクチャーが Information Server 製品モジュールおよびビジネス・ユース・ケース・シナリオのそれぞれに適さない場合があります。

Information Server のデプロイメント・アーキテクチャーを選択する際、さまざまな要因を考慮する必要があります。Information Server 製品モジュールを効果的に使用してビジネス目標を達成するために、本章では、以下の 4 つのデプロイメント・アーキテクチャーを紹介します。

- シンプル
- ユニファイド
- レポーティング
- ガバナンス

後続のセクションでは、お客様の状況に適合する可能性があるアーキテクチャー、実装ステップ、および各アーキテクチャーのメリットとデメリットについて説明します。

5.1 「シンプル」アーキテクチャー

「シンプル」は、標準的な開発環境、テスト環境、本番環境のアーキテクチャーです。標準的なデプロイメント・アーキテクチャーでは、ソース・コードは開発環境で開発され、テスト後にテスト環境にデプロイされ、再テスト後に本番環境にデプロイされます。このプロセスは一般に何かしらのソース・コード管理ツールを使用して行われます。本番環境での主プロセスがデプロイメントの影響を受けることが決してないよう、本番環境へのデプロイメントは本番稼働の管理とスケジューリングに従って行われます。

DataStage および QualityStage の開発は、この標準的な開発ライフサイクルに適合します。重要な考慮事項は、本番環境が他のプロセスによる悪影響を受けてはならないことです。この場合の本番環境の主目的は*データ統合*であるためです。

その他の Information Server 製品モジュール (IBM Information Analyzer (IA)、FastTrack、Business Glossary、および Metadata Workbench など) を DataStage および QualityStage と共に使用する場合、本番環境の統合プロセスとは別に、これらの他のモジュールのために別途新たにリソースを割り当てる必要があるかどうかを検討する必要があります。

選択するアーキテクチャーの決定

DataStage または QualityStage のみを使用している場合、一般的にはこの「シンプル」アーキテクチャーを選択します。Information Services Director は、DataStage ジョブと同じように処理されます。

その他の Information Server 製品モジュール (Information Analyzer、FastTrack、Business Glossary、および Metadata Workbench など) を使用している場合でも、この「シンプル」アーキテクチャーが適切なアーキテクチャーとなる場合があります。その場合は、この「シンプル」アーキテクチャーが適切であるかどうかを判断するために、以下の質問を検討します。

- いずれかの Information Server 製品モジュールで生成されたメタデータを他の目的で再使用しますか。コラボレーションはありますか。

  以下に例を挙げます。

  - Information Analyzer のプロファイリング結果をパブリッシュし、DataStage のジョブ開発で再利用する
  - IT 資産をビジネス用語にリンクする
- 本番環境でのデータ統合ジョブの実行は、以下のいずれかの処理の影響を受けていますか。
  - Metadata Workbench を使用したメタデータのレポーティング
  - Business Glossary を使用したグロッサリーの共有
  - Information Analyzer でのデータ・プロファイリング、または品質ルールの使用 (DataStage ジョブでデータ統合のために使用されるルール・ステージは、DataStage ジョブの一部であるため、含まれません)

  本番環境の主目的はデータ統合であるため、他のどのプロセスの影響も受けてはなりません。

お客様の状況に上記のいずれかの点が該当する場合、「シンプル」アーキテクチャーは、おそらく、お客様にとって正しいアーキテクチャーの選択肢*ではありません*。「ユニファイド」または「レポーティング」のいずれかのアーキテクチャーを検討してください。

標準的な構成

図 5-1 に示すように、この「シンプル」アーキテクチャーの標準的な構成は開発、テスト、本番の 3 層構成です。

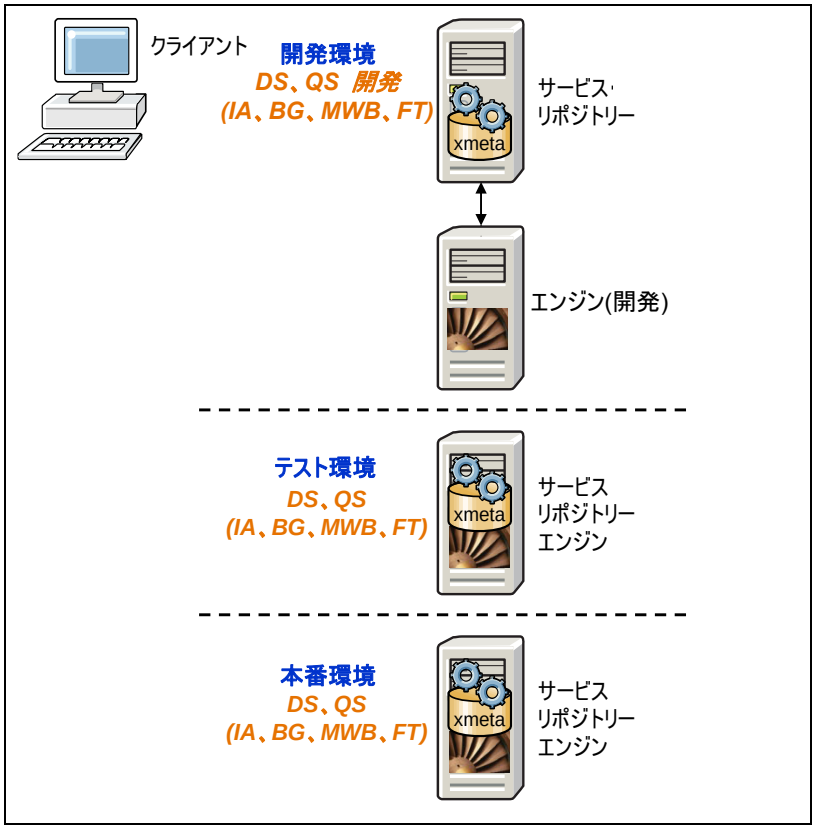


図 5-1　「シンプル」アーキテクチャーの 3 層構成

トポロジー

開発環境では、3 層構成が一般的です。サービス層とエンジン層のリソースを分離することにより、開発者がエンジン層で必要とするリソースの最適化および柔軟なスケーラビリティーを可能にし、また同時にサービス層は独立して負荷を管理できます。

テストおよび本番環境では、大半のお客様は標準の 2 層構成を使用します。2 層構成の詳細については、2.4『クライアント / サーバー・インストール・トポロジー』（21 ページ）および 2.7『Information Server クラスター・インストール』（24 ページ）で説明しています。

実装方法

図 5-2 に、3 層構成のデプロイメント・プロセスの例を示します。

DataStage および QualityStage のジョブは、標準的な開発、テスト、および本番の開発ライフサイクル・モデルに従います。以下のステップは、基本的なプロセスです。

1. ジョブは開発環境で、開発およびテストされます。
2. ジョブはテスト環境にデプロイされた後、テストされます。
3. ジョブは 本番環境にデプロイされます。

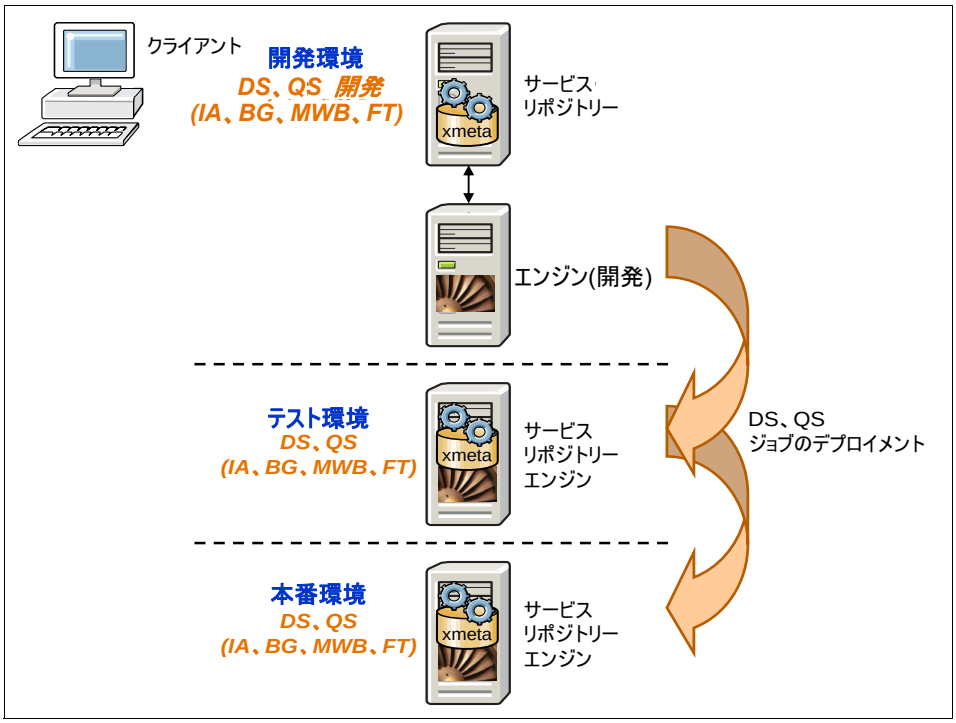


図 5-2　3 層構成のデプロイメント・プロセス

メリットとデメリット

この「シンプル」アーキテクチャーにはメリットとデメリットがあります。

メリット

生成されるすべてのメタデータ (テクニカル、オペレーショナル、ビジネス) は既に本番環境に存在するため、メタデータを別の環境にコピーする必要はありません。

デメリット

メタデータのレポーティングまたはデータ・プロファイリングが最小量を超えると、本番環境のデータ統合プロセスに影響を与える可能性があります。

「ユニファイド」アーキテクチャー : 本番環境のデータ統合が他のプロセスの影響を受ける場合は、「ユニファイド」アーキテクチャーの使用を検討してください。

5.2 「ユニファイド」アーキテクチャー

Information Server の Information Server 共有リポジトリーがコラボレーションとすべてのメタデータ (オペレーショナル・メタデータおよびビジネス・メタデータを含む) のレポーティングに使用されます。「ユニファイド」アーキテクチャーは、開発環境と、メタデータ・レポーティングおよびコラボレーション環境の組み合わせです。

この「ユニファイド」アーキテクチャーを確立する重要な理由は、共有されるビジネス・メタデータ (プロファイリングおよびビジネス・ルールなどのために) が必ずしも、4.24『その他の資産のライフサイクル管理』（134 ページ）に示す標準のソフトウェア開発ライフサイクルに従わないことです。例えば開発関連の作業をサポートするために一部の操作 (プロファイリングなど) を使用することもできます。また、機密保護の要件によって処理エンジンの分離が求められる可能性はありますが、依然としてメタデータを共有する必要があります (5.4『「ガバナンス」アーキテクチャー』（154 ページ）を参照)。

コラボレーションとレポーティングは最小限になりますが、メタデータ製品モジュールが本番環境に存在する場合、「シンプル」アーキテクチャーが最も推奨される選択肢となります。ただし、課題は残ります。メタデータのレポーティングとコラボレーションの機能をプロビジョンするのに最適な環境はどれでしょうか。メタデータに基づくレポーティングの「本番」という性質を考慮すると、(「シンプル」デプロイメント・シナリオと同様に) 本番環境を検討するでしょう。ただし、本番環境の主目的は、ダウンストリームのアプリケーションで使用される予定の「データ」を統合することです。Business Glossary の頻繁な使用、データ品質ルールの定期的な処理、リネージュおよび影響分析のレポート生成は、本番環境の「データ」パラダイムに適合しません。さらに、本番環境には「コラボレーション」の相手となる開発者はいません。したがって、このユース・ケースの場合は、これらの他の製品モジュールをデプロイする別の環境を選択することが必要です。

最も有益な結果は、開発環境の位置づけをデータ品質ルールの処理を考慮したものに変更することによって達成できます。開発環境において、開発を主目的としたエンジンから完全に分離された別の処理エンジンで本番データを使用することで、データのプライバシーとセキュリティーを確保できます。このように、*以前の* 開発環境は複数の目的に対応すると同時に、引き続き、機能要件と非機能要件がそれぞれ異なる各目的にも適合します。この状況では、この環境は複数の目的に対応しているため、もはや「シンプル」アーキテクチャーではなく「ユニファイド」アーキテクチャーになっています。「ユニファイド」アーキテクチャーは、コラボレーション、より幅広いデータ、そしてメタデータ・レポーティングをサポートするものです。

この「ユニファイド」アーキテクチャーを作成することにより、本番環境を主目的の *データ統合* の専用にすることができ、他の機能の影響を受けないようにすることができます。

選択するアーキテクチャーの決定

Information Analyzer、FastTrack、Business Glossary、および Metadata Workbench を DataStage、QualityStage、および Information Services Director と共に使用している場合、一般的な手法では、この「ユニファイド」アーキテクチャーを検討します。

ただし、これらの製品モジュールを使用している場合でも、他のデプロイメント・アーキテクチャーの方が適していることがあります。例えば、これらの製品モジュールに専用のリソースや開発者とのコラボレーションが必要でない場合は、「シンプル」アーキテクチャーの方が適しています。膨大な量のレポーティングの要件がある場合は、「レポーティング」アーキテクチャー (5.3『「レポーティング」アーキテクチャー』（150 ページ）を参照) の方が適している可能性があります。

標準的な構成

図 5-3 に示すように、このアーキテクチャーの標準的な構成は、ユニファイド / 開発、テスト、および本番の 3 層環境です。

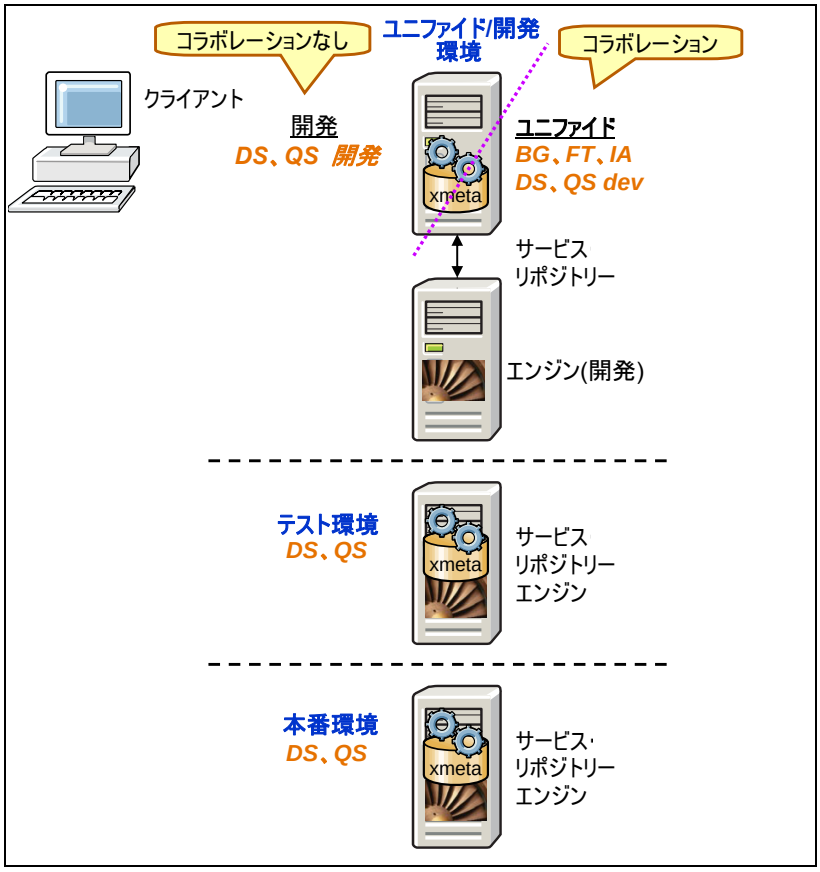


図 5-3 「ユニファイド」アーキテクチャーの 3 層環境

「ユニファイド」アーキテクチャーでは、開発環境は、DataStage および QualityStage ジョブの開発だけに使用されるのではなく、メタデータ・レポーティングおよびコラボレーションの環境としても使用されます。開発環境が複数の目的に対応している点が、「シンプル」アーキテクチャーと「ユニファイド」アーキテクチャーの主な違いです。この理由から、開発環境が ユニファイド / 開発環境に変わっています。

テストおよび 本番環境は、DataStage および QualityStage のジョブの実行環境としてのみ有用です。

トポロジー

ユニファイド / 開発環境では、開発環境と同様に 3 層構成が一般的です。その理由は、多くの同時ユーザーによるコンパイルとテストのためにリソースが必要であるためです。Information Server リポジトリーの下に複数のエンジンを配置することもできます。これにより、DataStage および QualityStage の開発を共通の共有メタデータ・プロセスから分離できます。

Information Analyzer を使用する場合、図 5-4 に示すように、Information Analyzer に専用のエンジンを配置すると有用です。

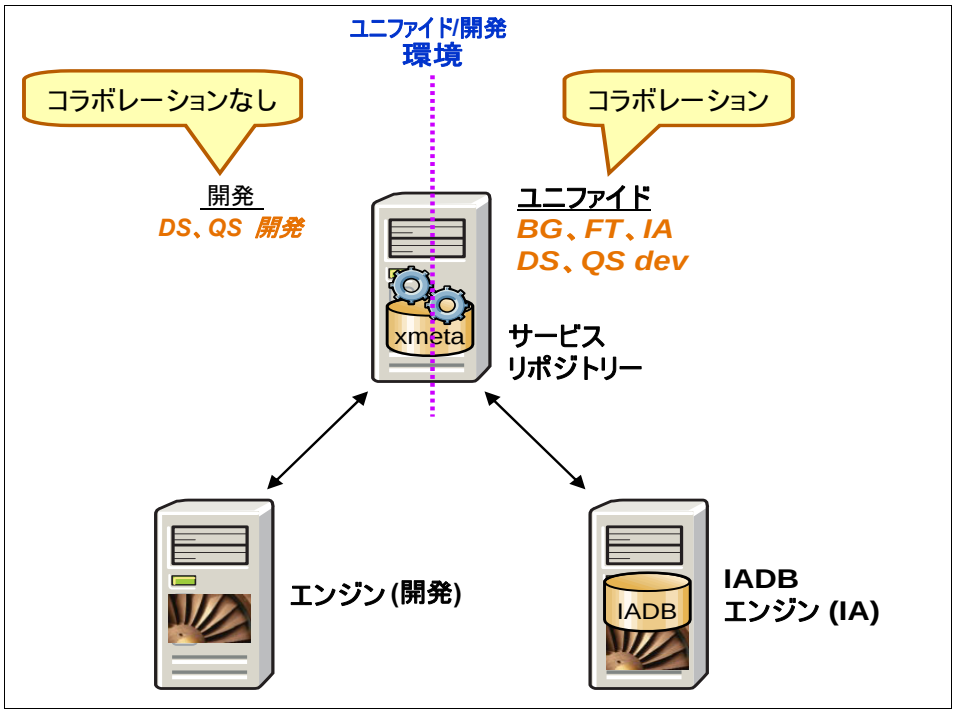


図 5-4　Information Analyzer を使用する「ユニファイド」アーキテクチャー

専用のエンジンにより、安全な方法で本番データに接続してデータを処理することができます。各エンジン・ノードの用途は次のとおりです。

- 開発エンジン・ノード : このエンジン・ノードを使用して、開発者は、そのサーバーでジョブの開発、コンパイル、およびテストを実行できます。
- Information Analyzer エンジン・ノード : このエンジン・ノードを使用して、本番データを接続することにより、データおよびデータ品質ルールのプロファイリングが行われます。スタンドアロン・データベース (IADB) があるため、データを保護すると同時に、サンプリングされたデータに関するパフォーマンス上の考慮事項に対応することもできます。データの保護に関する詳細は、『機密データのデータ・プロファイリング』（146 ページ）で説明しています。

テストおよび 本番環境では、大半のお客様が、セキュリティー、用途、および可用性の理由から標準の 2 層構成を使用します。

実装方法

図 5-5 (145 ページ) に、実装のイメージを示します。基本的な実装のステップは次のとおりです。

- DataStage および QualityStage のジョブは、標準的な開発、テスト、および本番の開発ライフサイクル・モデルに従います。
- メタデータ・レポーティングの目的で、DataStage および QualityStage のジョブデザインとオペレーショナル・メタデータ、および外部メタデータを「ユニファイド」アーキテクチャー環境にコピーする必要があります。
  - DataStage および QualityStage のジョブデザインメタデータはユニファイド / 開発環境で生成されます。ただし、引き続きデザイン・メタデータをコピーし、同じ命名規則を使用して 開発環境とユニファイド / 本番環境との間で現行の 本番環境のバージョンを一致させる必要があります。

> **ジョブデザイン :** DataStage ジョブで使用される FastTrack マッピングおよび Information Analyzer ルール・ステージは、このジョブデザインに含まれます。

– DataStage および QualityStage のオペレーショナル・メタデータは、ジョブ実行時にファイル・システムに保管される XML ファイルとして生成されます。本番環境のエンジン層に配置されているシェル・スクリプト (RunImportStart.sh) またはバッチ・ファイル (RunImportStart.bat) を使用して、このオペレーショナル・メタデータを xmeta にインポートできます。メタデータをターゲットの xmeta にロードするための資格情報と接続はプロパティー・ファイル (runimport.cfg または runimport.cfg.unix) で指定されるため、ターゲットの xmeta に対して 本番環境ではなくユニファイド環境を指定する必要があります。その後、オペレーショナル・メタデータはユニファイド環境にロードされます。オペレーショナル・メタデータの管理の詳細については、次の場所にある Information Server 資料「*IBM InfoSphere Information Server, Guide to Managing Operational Metadata*」で説明しています。

  http://publibfp.boulder.ibm.com/epubs/pdf/c1934770.pdf

  解決 : デザイン・メタデータがメタデータ・レポーティングに十分でない場合、オペレーショナル・メタデータが有効です。この状況は、ジョブで適切なデフォルト値を指定せずにパラメーターを使用した場合によく起こります。オペレーショナル・メタデータは実行で使用された値を解決するため、メタデータを手動で切り替える必要はありません。詳細については、3.2.3『オペレーショナル・メタデータ』（54 ページ）を参照してください。

– PL/SQL および COBOL プログラムなどの外部資産をメタデータ・レポーティングに組み込む要件がある場合、ユニファイドの Information Server リポジトリーでその外部メタデータも生成する必要があります。

► ビジネス・メタデータはユニファイド環境で生成および使用されるため、ビジネス・メタデータをプロモートする必要はありません。

機能 : Business Glossary には、組み込み機能の一部として独自のワークフローがあります。したがって、ソフトウェア開発と同じ SDLC の考慮事項は必要ありません。

► Information Analyzer のプロファイリング結果はユニファイド環境で生成およびパブリッシュされるため、プロモートする必要はありません。

ライフサイクル : Information Analyzer には、標準的なものとは異なる独自の開発ライフサイクル機能があります。プロファイリングの結果はパブリッシュされて、潜在的なデータの変則性とデータ・フォーマットやそれらのコーディング方法などの概念を理解する必要がある DataStage 開発者などの他の製品モジュールのユーザーに対して表示できるようになります。

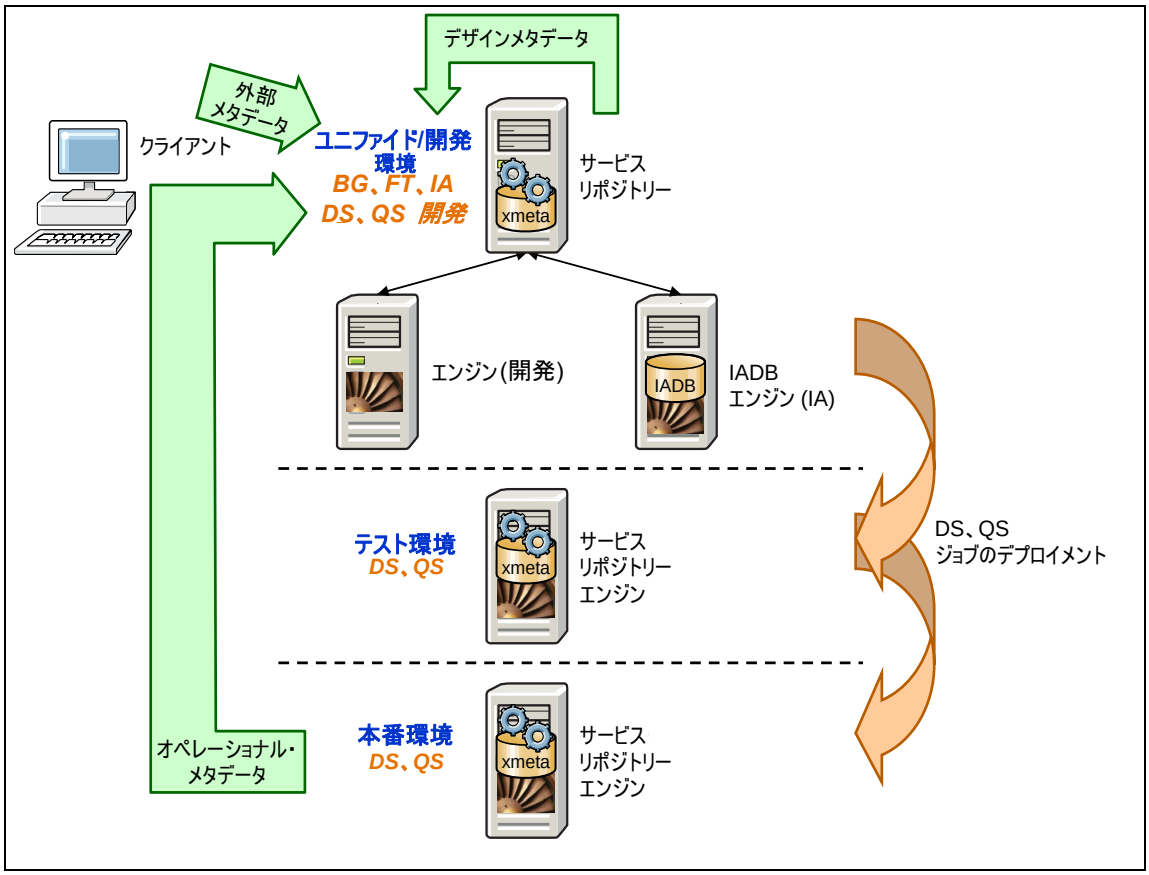


図 5-5　テストおよび本番環境の実装のイメージ

メリットとデメリット

このユニファイド・アーキテクチャーにはメリットとデメリットがあります。

メリット

以下のメリットが該当します。

- メタデータ・レポーティングは別個の環境で行われるため、各種メタデータのニーズによる 本番環境への影響は最小限に抑えられます。
- すべての製品モジュールが同じ環境に配置されているため、メタデータを再活用でき、コラボレーションはシンプルになります。例えば、ビジネス・メタデータは、Business Glossary と、ユニファイド / 開発環境の作成者やユーザーの両方によって生成および使用されます。
- Information Analyzer には個別のエンジンがあるため、Information Analyzer のプロファイリングによる 開発環境への影響は最小限に抑えられます。
- Information Analyzer エンジンおよび IADB は 開発環境から分離されているため、本番データは安全です。
- ユニファイド / 開発環境は 3 層構成であるため、メタデータのニーズと開発により負荷を改善するために、さらに多くのサーバー層を追加できます。3 層構成の詳細については、17 ページの第 2 章、『Information Server のインストール・トポロジー』で説明しています。

デメリット

以下のデメリットが該当します。

- Metadata Workbench/Business Glossary を使用するユーザーが多数の場合や、グロッサリーが複雑である場合は、開発環境のパフォーマンスに影響を与える可能性があり、その逆も考えられます。

解決方法 : レポーティング専用に新しく個別のレポーティング環境を作成することでこの問題を解決できます。5.3『「レポーティング」アーキテクチャー』（150 ページ）を参照してください。

- オペレーショナル・メタデータについては、メタデータが別の環境で生成されたという事実を反映するためにプロパティー・ファイルを若干変更しなければならない場合があります。

ヒント : このような環境では、十分に理解されている変更管理プロセスを整備する必要があります。

- オペレーショナル・メタデータをユニファイド環境にコピーするために、さらに多くの作業が必要になります。

ヒント : ユニファイド環境の Information Server リポジトリーはすべてのメタデータを共有するため、他のすべ てのユーザーが使用している共有環境に影響を与えずにパッチの適用やテスト、またテスト・ケース を実行できる環境が必要です。この環境は、スタンドアロンの「サンドボックス」環境であるべきです。

機密データのデータ・プロファイリング

データ・プロファイリングは、一般に本番データに対して行われます。情報およびデータ・セキュリティーの要件では、この本番データへのアクセスに対する厳格な制御が求められます。本番実行環境を分離すると、この動作を保証できます。ユニファイド環境では、セキュリティー・ルールに対応するために実際の本番データと詳細な値が分離される一方で、この分析の構造的結果 (メタデータおよびフォーマットなど) は、ユニファイドの Information Server リポジトリーで DataStage および QualityStage のユーザーと共有されます。

エンジン・ノードの分離

図 5-6 に示すように、エンジンは別のサーバー上で Information Analyzer 専用となっています。これにより、本番データは、開発環境、開発者、および開発データから完全に分離されます。さらに、『ユーザー認証およびロール』（148 ページ）で説明している Information Server の組み込みセキュリティー機能により、ユーザーは、Information Server の製品モジュールのロールおよび処理を許可されている製品モジュール内のプロジェクトから権限を得ます。ただし、エンジン・ノードと IADB データベースを DataStage または QualityStage 環境からファイル・システム・レベルでも分離することによって、さらにセキュリティーを強化できます。

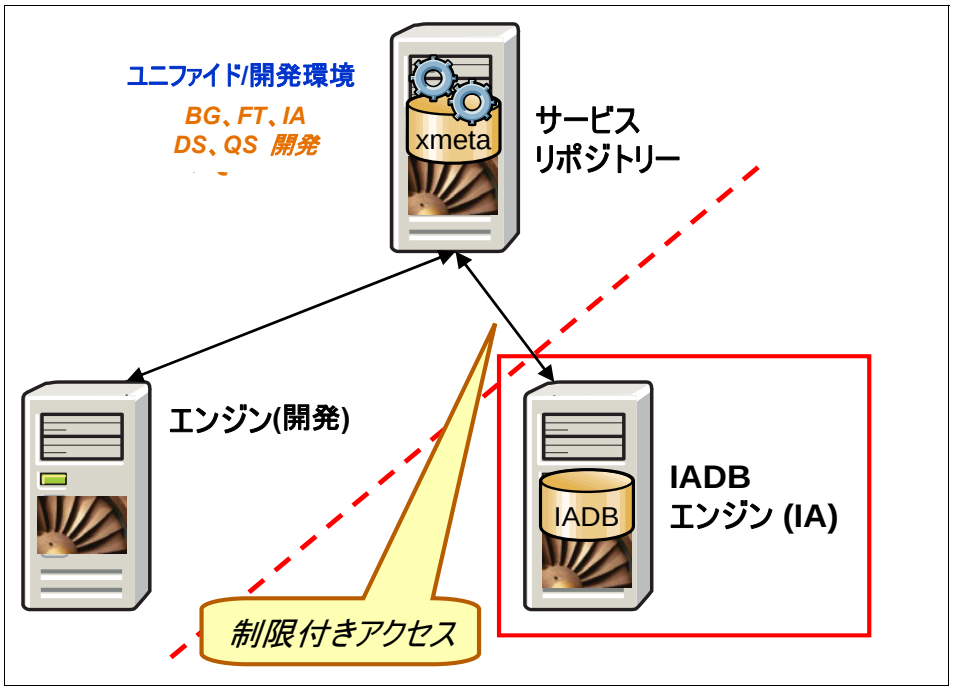


図 5-6　スタンドアロン・サーバー上の IADB 搭載の IA エンジン

本番データを分析する前の時点で、非本番データが入った IA 環境が必要となる場合があります。このソース・データを作成するには、必要に応じて、本番環境から非機密データをプロモートするか、機密データをマスキングします。Information Analyzer では、プロジェクトごとに別の IADB を指定できるため、適切な許可を持つ別のノードまたはファイル・システムで IADB を作成することで、さまざまなタイプのデータが入った環境をそれぞれ分離できます。

図 5-7 は、Information Analyzer プロジェクトで IADB を指定する例です。

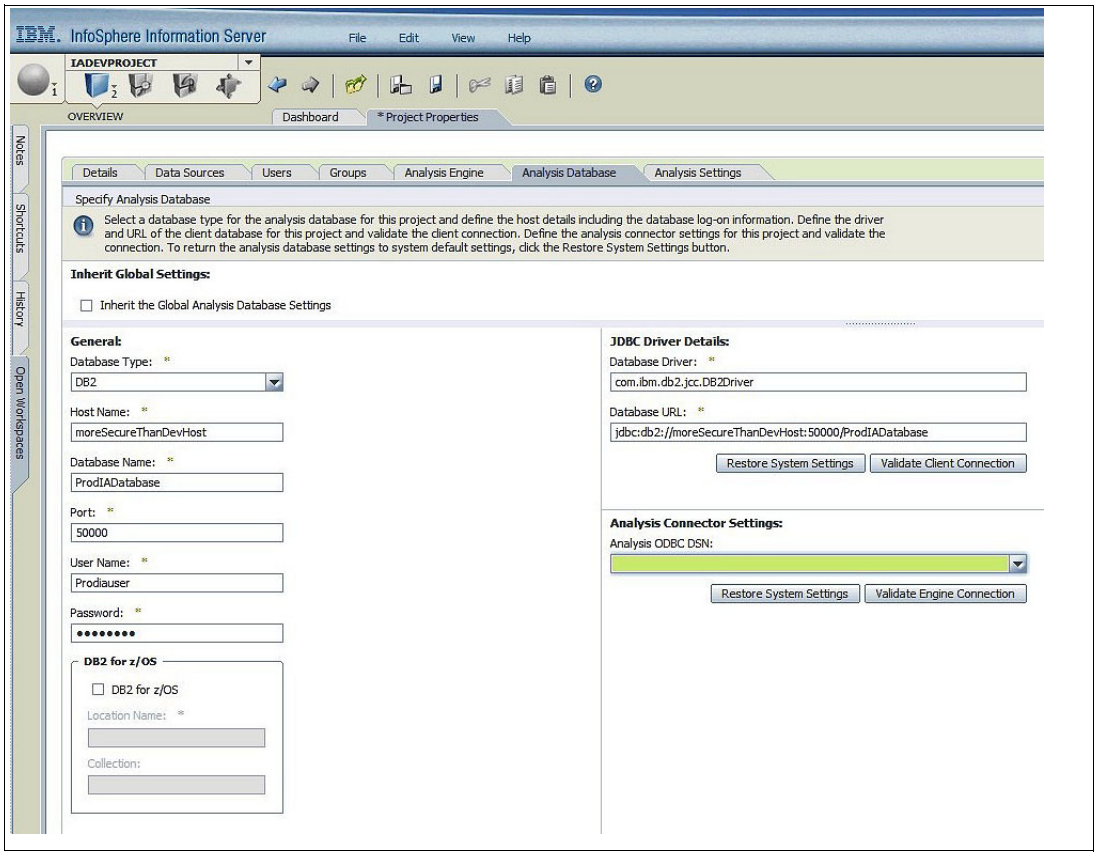


図 5-7　プロジェクトごとの IADB の指定

ユーザー認証およびロール

Information Server は、ロール・ベースのアクセス制御をサポートします。ユーザーは、Information Server のロールの共用体、スイート・コンポーネント (DataStage および Information Analyzer など) のロール、および作業しているプロジェクトから権限を得ます。

図 5-8 に、Information Server のセキュリティー・ロールの概要を示します。

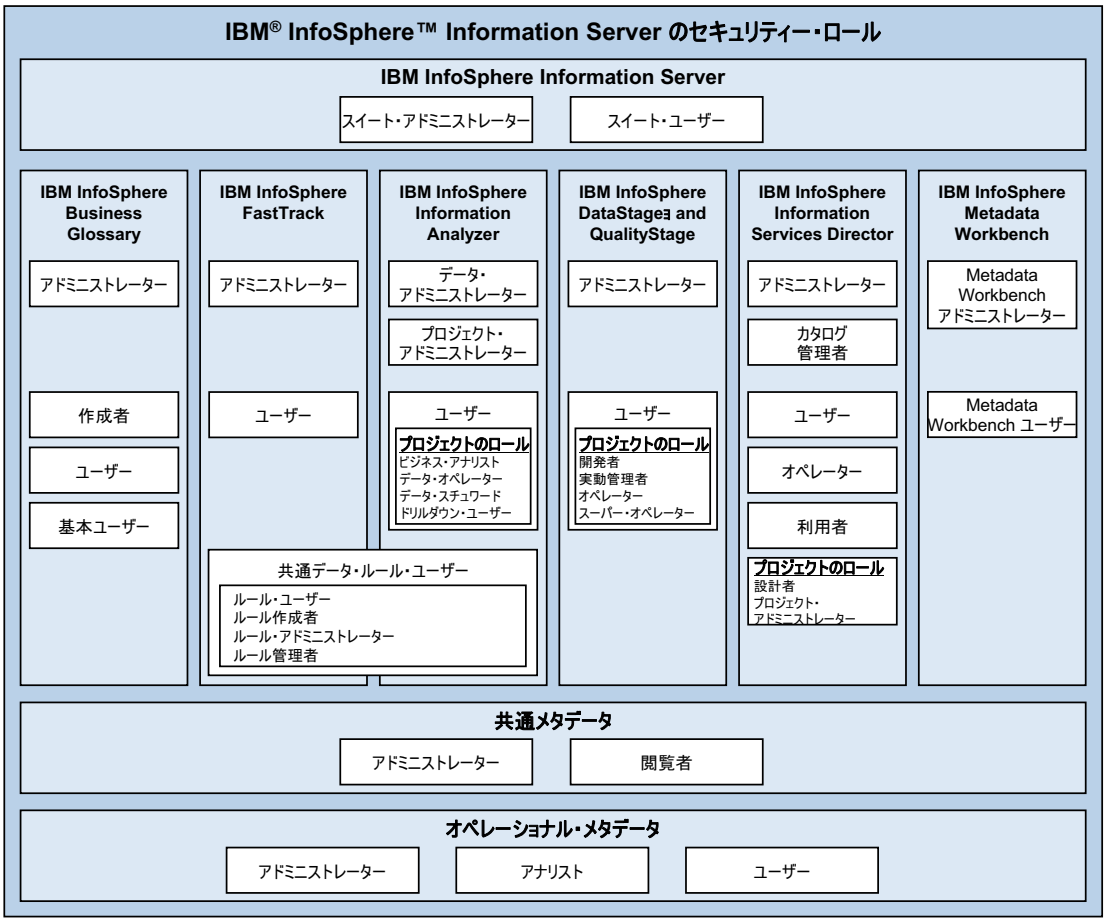


図 5-8 IA Server のセキュリティー・ロール

Information Analyzer と、DataStage および QualityStage で別々のロールを持つ別々のユーザーを作成することにより、各製品へのアクセスを制御できます。プロジェクト・レベルでも認証を設定できるため、Information Analyzer に複数のプロジェクトが存在する場合にアクセスを制御できます。ユーザー・ロールの詳細については、次のインフォメーション・センターにある「*IBM InfoSphere Information Server Administration Guide*」(SC18-9929-05) で説明されています。

http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r7/topic/com.ibm.swg.im.iis.found.admin.common.doc/topics/c_wdisad_Roles.html

5.3 「レポーティング」アーキテクチャー

「レポーティング」アーキテクチャーには、Information Server 内に存在するさまざまなタイプのメタデータ (ビジネスおよびテクニカル) への常時アクセスをすべてのステークホルダーに提供するための専用のメタデータ環境があります。

この専用の「レポーティング」環境では、メタデータが作成および保持される環境に関係なく、関連するメタデータは、Information Server のこの別個のインスタンスにコピーされます。これは、メタデータを蓄積するための専用のインスタンスです。

このアーキテクチャーでは、システムは、他の既存の環境に影響を与えることなく、Business Glossary の表示、データ品質ルールの処理、または Metadata Workbench レポートの生成といった多数のユーザー要求を処理できます。

選択するアーキテクチャーの決定

Information Analyzer、FastTrack、Business Glossary、および Metadata Workbench を DataStage、QualityStage、および Information Services Director と共に使用していて、ユニファイド / 開発環境への影響が大きくて要件をサポートできない場合は、この「レポーティング」アーキテクチャーの使用を検討してください。要件として、多数のユーザー (100 以上など) が Metadata Workbench または Business Glossary を使用していることや、グロッサリーが複雑であることが挙げられます。

標準的な構成

図 5-9 (151 ページ) に示すように、このアーキテクチャーの標準的な構成は、ユニファイド / 開発、テスト、本番、レポーティングの 4 つの環境から成る構成です。

使用されている製品モジュールと、それらの間のコラボレーションによっては、ユニファイド / 開発がシンプルな 開発環境となる場合があります。例えば、Information Analyzer のデータ品質のルール機能が使用されていて、ビジネス・アナリストがビジネスへの報告のために結果を確認する場合は、Information Analyzer によって生成されたメタデータは他の目的で再使用されません。この場合は、他の製品モジュールとのコラボレーションがないため、ユニファイド環境は必要ではなく、環境は単純な 開発環境となります。

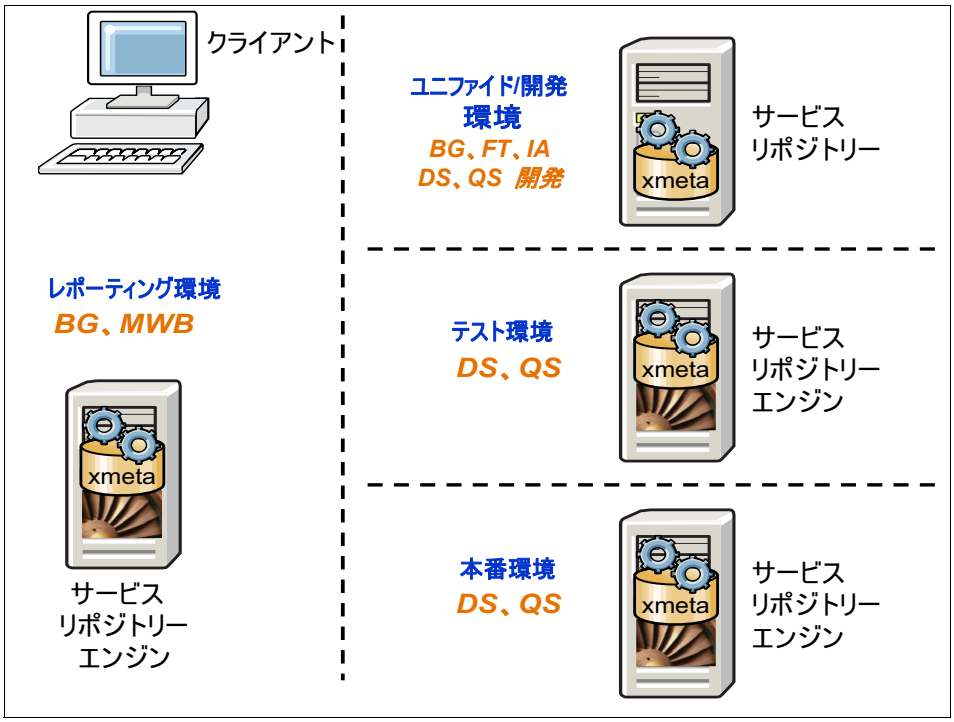


図 5-9　標準的なアーキテクチャー構造

トポロジー

ユニファイド / 開発または 開発環境では、3 層構成が一般的です。その理由は、多くの同時ユーザーのコンパイルとテストがジョブ開発者のパフォーマンスに影響を与える可能性があるためです。

テストおよび 本番環境では、大半のお客様が、セキュリティー、用途、および可用性を考慮して標準の 2 層構成を使用します。

レポーティング環境では、2 層構成も一般的です。ただし、レポーティングの要件によっては、17 ページの第 2 章、『Information Server のインストール・トポロジー』で説明しているようにサービス層を拡張できるため、3 層構成も適している場合があります。また、既存の環境 (開発、テスト、本番) への別のエンジンの追加を選択する場合も 3 層構成になります。

実装方法

図 5-10 (153 ページ) に、実装のイメージを示します。基本的な実装のステップは次のとおりです。

- DataStage および QualityStage のジョブは、標準的な開発、テスト、および本番の開発ライフサイクル・モデルに従います。
- メタデータ・レポーティングの目的で、レポーティング環境に DataStage および QualityStage のジョブデザインとオペレーショナル・メタデータをコピーして、外部メタデータをインポートする必要があります。
  - DataStage/QualityStage のジョブデザインメタデータはユニファイド / 開発環境で生成されます。そのため、このメタデータをレポーティング環境にコピーする必要があります。

> **設計 :** DataStage ジョブで使用される FastTrack マッピングおよび Information Analyzer ルール・ステージは、このジョブデザインの一部です。このステップの実行には、51 ページの第 3 章、『メタデータ管理』で説明しているいずれのツールも使用できます。

– DataStage または QualityStage のオペレーショナル・メタデータは、ジョブ実行時にファイル・システムに保管される XML ファイルとして生成されます。本番環境のエンジン層に配置されているシェル・スクリプト (RunImportStart.sh) またはバッチ・ファイル (RunImportStart.bat) を使用して、このオペレーショナル・メタデータを xmeta にインポートできます。メタデータをターゲットの xmeta にロードするための資格情報と接続はプロパティー・ファイル (runimport.cfg または runimport.cfg.unix) で指定されるため、ターゲットの xmeta に対して 本番環境ではなくレポーティング環境を指定する必要があります。すると、オペレーショナル・メタデータはレポーティング環境にロードされます。オペレーショナル・メタデータの管理の詳細については、次のアドレスにある「*IBM InfoSphere Information Server, Guide to Managing Operational Metadata*」で説明されています。

  http://publibfp.boulder.ibm.com/epubs/pdf/c1934770.pdf

  値 : デザイン・メタデータがメタデータ・レポーティングに不十分である場合、オペレーショナル・メタデータが必要です。この状況は、ジョブで適切なデフォルト値を指定せずにパラメーターを使用した場合によく起こります。オペレーショナル・メタデータには実行で使用された値が指定されるため、メタデータを手動で切り替える必要はありません (51 ページの第 3 章、『メタデータ管理』を参照)。

– PL/SQL および COBOL プログラムなどの外部資産をメタデータのレポーティングに組み込む要件がある場合、それらの外部メタデータをレポーティングの Information Server リポジトリーにインポートすることも必要です。51 ページの第 3 章、『メタデータ管理』で説明しているように、これは多数の方法で実行できます。

► ビジネス・メタデータをユニファイド / 開発環境からレポーティング環境にコピーする必要があります。

ビジネス・メタデータは、ユニファイド / 開発環境で Business Glossary の作成者によって生成されますが、レポーティング環境で Business Glossary のユーザーによって使用されます。そのため、グロッサリーをレポーティング環境にインポートする必要があります。

Business Glossary: Business Glossary が他の製品モジュールとのコラボレーションで使用されていない場合は、グロッサリーの生成と使用の両方をレポーティング環境で行うことができます。

Business Glossary には、DataStage および QualityStage とは異なり、標準の開発 / テスト / 本番開発サイクルには従わない独自の開発ライフサイクル機能があります。

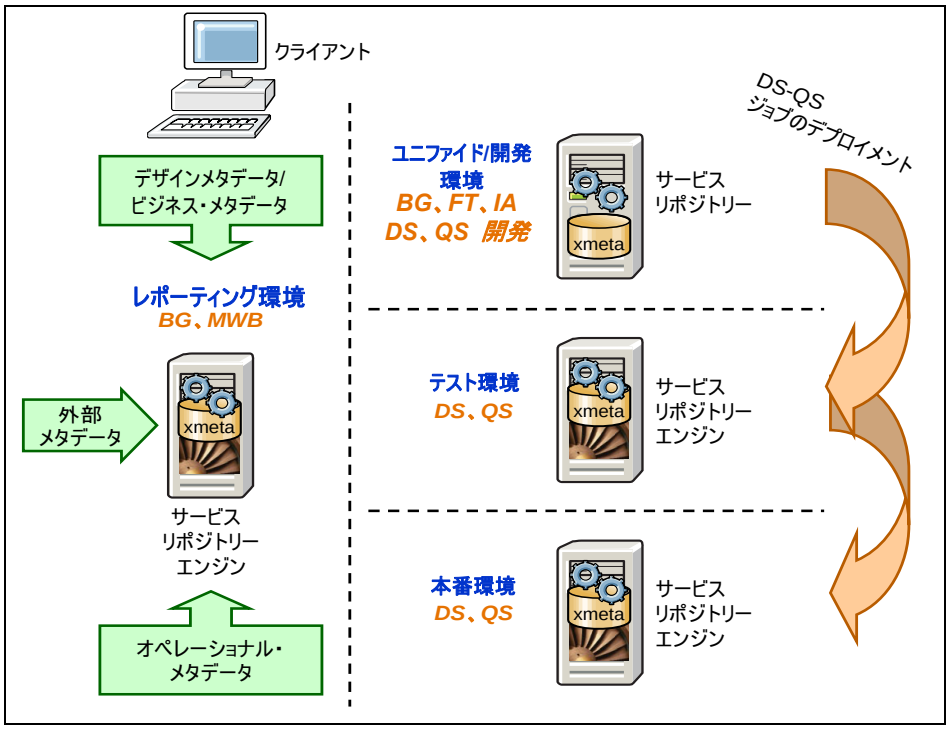


図 5-10　実装のイメージ

メリットとデメリット

この「レポーティング」アーキテクチャーにはメリットとデメリットがあります。

メリット

以下のメリットが該当します。

- メタデータ・レポーティングは専用のレポーティング環境で行われるため、本番 / テスト / 開発環境はさまざまなメタデータのニーズの影響を受けません。
- Information Analyzer には個別のエンジンがあるため、Information Analyzer のプロファイリングによる開発環境への影響は最小限に抑えられます。
- Information Analyzer エンジンおよび IADB が 開発環境から分離されるため、本番データは安全です。

デメリット

以下のデメリットが該当します。

- オペレーショナル・メタデータについては、メタデータが別の環境で生成されたという事実を反映するためにプロパティー・ファイルを若干変更しなければならない場合があります。

  変更管理プロセス：このような環境では、十分に理解されている変更管理プロセスを整備する必要があります。

- メタデータをレポーティング環境にコピーするために、さらに多くの作業が必要になります。
- このレポーティング環境には、追加のハードウェアおよびソフトウェアが必要です。

オプション : レポーティング環境ではジョブは実行されませんが、ジョブデザインをロードするために DataStage エンジンが必要です。レポーティングはジョブの実行に影響を与えないため、ライセンスの観点から、いずれかの既存の環境 (開発、テスト、本番) への別のエンジンの追加が候補として考えられます。この場合、ジョブのコンパイルなどのサービス層への負荷の高い処理による影響を考慮してください。オペレーショナル・データの使用の点では、本番環境のインストール・タグ (itag) の使用は利点があります。すなわち、メタデータのレポーティングのためにホスト名を変更する必要がありません。

5.4 「ガバナンス」アーキテクチャー

DataStage または QualityStage を使用しておらず、Information Server のメタデータ・レポーティングまたはガバナンスの機能を使用したい場合、このスタンドアロンの「ガバナンス」環境を作成できます。

この場合、外部メタデータを分析および報告でき、企業のグロッサリーを作成または共有して企業のガバナンスを強化することができます。

Information Analyzer または FastTrack を使用していて、他のモジュールとのコラボレーションがない場合にも、この環境は適しています。該当する環境の例については、『選択するアーキテクチャーの決定』（154 ページ）で説明しています。

選択するアーキテクチャーの決定

この環境で使用される主な製品モジュールは、Metadata Workbench および Business Glossary です。

使用している機能と、複数の製品モジュール間のコラボレーションがあるかどうかに応じて、この環境で Information Analyzer および FastTrack を使用することも適しています。DataStage または QualityStage も使用している場合は、本章で説明している他の 3 つのアーキテクチャーのいずれかの方が適しています。

以下のケースでは、Information Analyzer および FastTrack の機能は、他の製品モジュールとのコラボレーションがない単一製品として使用されています。その他のケースもあります。

- Information Analyzer のプロファイリング機能がデータ品質評価に使用されます。この場合、ビジネス・アナリストは、結果を確認して、ビジネス、ソース・システム所有者、スチュワードなどに報告しますが、結果として生成されるメタデータを他の目的に再使用しません。また、他の製品モジュールとのコラボレーションはありません。
- Information Analyzer のデータ品質ルール機能は、(DataStage のルール・ステージではなく) Information Analyzer の GUI を介して使用されます。
- FastTrack のマッピング仕様機能は、ガバナンスのために使用されますが、DataStage ジョブ・テンプレートの作成には使用されません。

標準的な構成

図 5-11 に示すように、このアーキテクチャーの標準的な構成はスタンドアロンの「ガバナンス」環境です。

図 5-11　スタンドアロン・ガバナンスの構成

エンジン : Information Analyzer を使用する場合を除き、エンジンは必要ありません。

トポロジー

「ガバナンス」環境の一般的なトポロジーは、2 層のスタンドアロン環境です。負荷に応じて、環境を拡張できます。拡張には複数のパターンがあります。詳細については、17 ページの第 2 章、『Information Server のインストール・トポロジー』で説明しています。

実装方法

図 5-12 (156 ページ) に、実装のイメージを示します。基本的なステップは次のとおりです。

- 51 ページの第 3 章、『メタデータ管理』で説明しているツールを使用して外部メタデータをインポートします。
- ビジネス・メタデータはメタデータ・レポーティング環境で生成および使用されるため、ビジネス・メタデータに関して何もコピーする必要はありません。

ライフサイクル : 63 ページの第 4 章、『Information Server 資産のライフサイクル管理』で説明したように、Business Glossary には、DataStage および QualityStage とは異なり、標準の 開発 / テスト / 本番開発ライフサイクルに従わない独自の開発ライフサイクル機能があります。

- FastTrack のマッピング仕様もメタデータ・レポーティング環境で生成および使用されるため、マッピングに関して何もコピーする必要はありません。

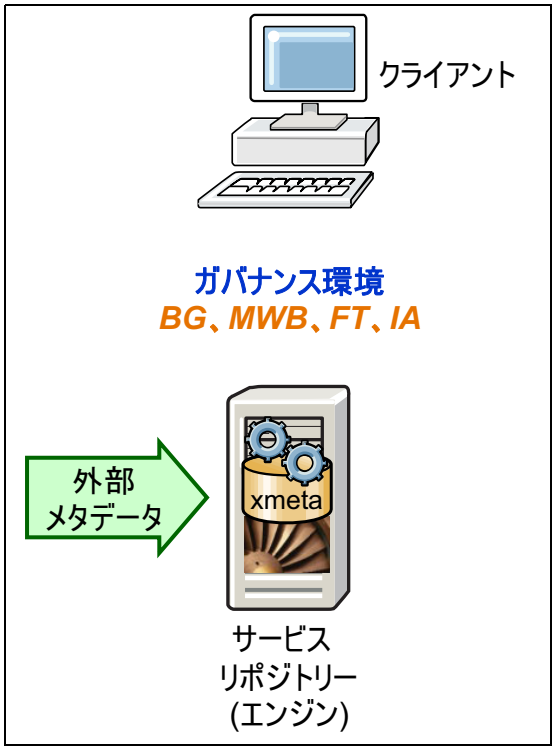


図 5-12　「ガバナンス」の実装のイメージ

5.5 デプロイするアーキテクチャーの選択

本章では、4 つのデプロイメント・アーキテクチャーと、それぞれのアーキテクチャーの考慮事項を紹介しました。それでは、どのようにしたら、特定の状況に応じてデプロイするアーキテクチャーを選択できるでしょうか。

アーキテクチャーを決定するための主な決定要因は次のとおりです。

- どの製品を使用していますか
- 複数の製品間でメタデータのコラボレーション (他の目的でのメタデータの再使用) はありますか
- リネージュや影響分析の実行、グロッサリーを使用するユーザーは何人ですか
- グロッサリーはどれくらい複雑ですか
- 環境にはどれくらいのリソースがありますか

図 5-13 のチャートは、お客様の状況に適切なアーキテクチャーを選択する上で役立ちます。

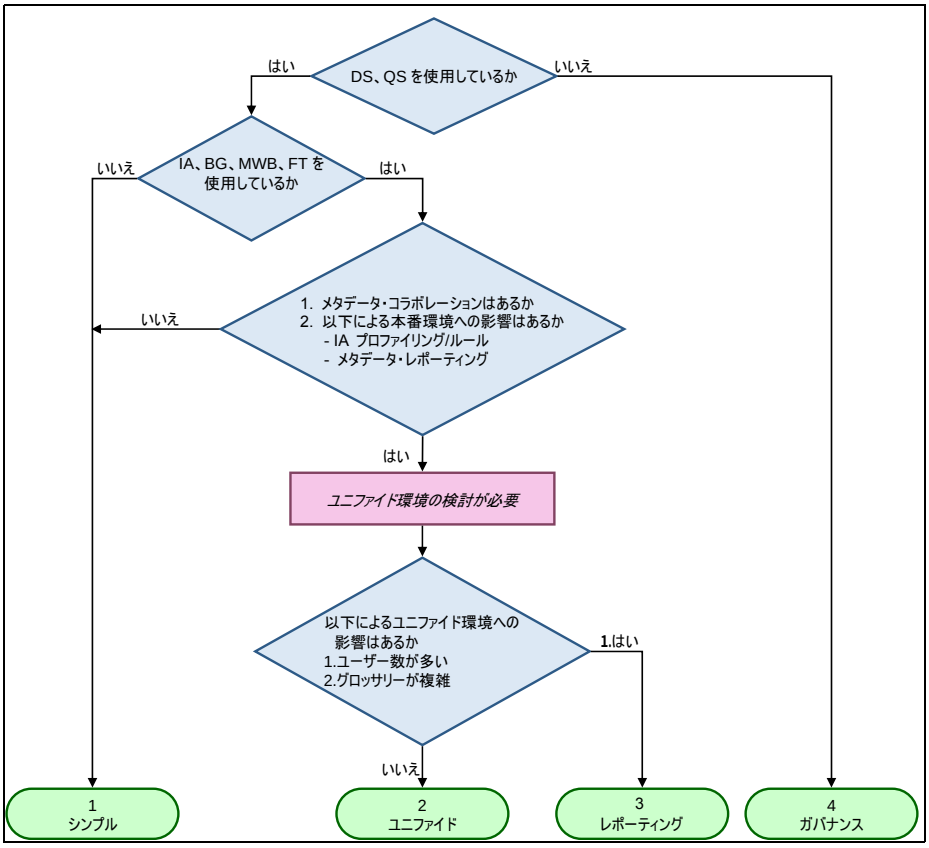


図 5-13　アーキテクチャーの選択を決定するためのフローチャート

6

デプロイメントのユース・ケース

本書では、最初に Information Server (IS) の概要、IS の製品モジュールの機能、サーバー・コンポーネントのインフラストラクチャー、アーキテクチャーのトポロジー、および IS のデプロイメントについて説明しました。また、関連する IS 製品モジュールを使用して開発されるソフトウェアの管理、ソフトウェア・ライフサイクル、そして究極的な目標である最終成果物の配信を通じたソフトウェア・デプロイメントの各段階をサポートするさまざまな環境についても説明しました。もう 1 つの目的は、共通理解を促進する上で役立つ概念に関する詳細情報を提供することです。

理解すべき重要な概念は、Information Server を利用するデータ統合プログラムから複数の「最終成果物」が生成されることです。代表的な最終成果物は*データ*です。このデータという成果物は、例えばビジネス・インテリジェンス、マスター・データ管理、データウェアハウスのためのアプリケーションで使用できる、再編成、クレンジング、および品質強化されたコンテンツという形になる場合があります。

ただし、Information Server を使用する場合、もう 1 つの最終成果物は*メタデータ*から生成されます。このメタデータという成果物は、ETL プロセスの副産物を超えるものであり、それ自体で多くの目的を果たします。データにビジネス上の意味を与え、ビジネスおよび情報技術 (IT) プロセスと情報に対するガバナンスを実現します。また、このデータがアプリケーションに与える影響、この情報を必要とするユーザー、この情報の作成と配信に必要なプロセス、データ・フローとソースからターゲットへの配信処理の検証に関するレポートも提供します。

お客様は、最終成果物としてデータのみを使用したり、メタデータの成果物のみを使用したり、多くの場合はデータとメタデータの両方の最終成果物を使用したりします。このように独特の「最終的成果物」の開発とライフサイクルはそれぞれ異なるため、本書で説明したいくつかのデプロイメント・アーキテクチャーの促進力となっています。

本章では、Information Server に基づくソリューションを実装するための、ビジネス要件がそれぞれ異なる 3 つのシナリオを検討します。本章では、これまでの章で説明した、適切なデプロイメント・アーキテクチャーを設計するための原則を踏まえています。この原則により、データ統合、データ・ガバナンス、マスター・データ管理といったプログラムの実装を成功させることができます。この例に基づく設計分析により、実装されるお客様固有のビジネス要件と実現する最終成果物を踏まえ、同じ原則を使用して適切な Information Server デプロイメント・アーキテクチャーの設計を成功させることができます。

6.1 メタデータの差異化要因

本書のこれまでの章では、インフラストラクチャーのコンテキストにおける Information Server、製品モジュールの機能、製品モジュールがインフラストラクチャーのどこに適合するか (IS 製品モジュールが相互に統合される方法など) について説明しました。統合の大半は、製品モジュール自体によって生成される内部メタデータと、さまざまな Information Server ツールを使用して IS 共有リポジトリーにインポートされる外部メタデータの両方の累積に基づいています。このメタデータ・コンポーネントは、通常、従来型のデータ統合プロセスとの重要な差異化要因であり、最適な方法で使用するためには新たな理解を必要とします。また、これらの章では、Information Server が膨大な数の複雑なビジネス要件をサポートすることを実証するために多数のデプロイメント・オプションを紹介しました。

この共有メタデータは、DataStage ジョブデザインのコードなどの開発されたソフトウェア・コードと同じ方法では管理または保守されません。例えば、Business Glossary (BG) 製品モジュールを複数の環境にデプロイすることは一般に必要ではありません。Business Glossary には独自の組み込みワークフローがあり、標準的な開発 / テスト / 本番ソフトウェア開発サイクルを必要としないためです。したがって、BG によって生成されるメタデータは、BG がデプロイされている環境でのみ保持されます。ただし、BG がユニファイド / 開発環境にデプロイされる場合 (137 ページの第 5 章、『メタデータのデプロイメント・アーキテクチャー』を参照)、同じ環境にデプロイされている他の製品モジュールによって使用される開発レベルの資産ではなく、資産の割り当てを目的として本番レベルの共通のテクニカル・メタデータが必要になる可能性があります。この例では、開発レベルと本番レベルの両方のメタデータが同じ環境に存在できます。

また、特定のデータ統合プロジェクトのために開発されるコードとライフサイクルと推定寿命は、必ずしもメタデータのライフサイクルと一致しません。最後に、*メタデータ*から得られるレポーティングは、必ずしもデータ統合開発プロセスからの*データ*から得られるレポーティングと同じ目的を果たすわけではありません。これらの問題が原因で、本章で分析するようにデプロイメント・アーキテクチャーが多様化しています。

6.2 ユース・ケース・シナリオ例

このセクションでは、3 つのユース・ケースをそれぞれのビジネス要件と共に示します。課題は、各シナリオの要件を分析して、本書で示した原則を適用し、ビジネス目標を達成するのに適切なデプロイメント・アーキテクチャーを決定することです。最初のシナリオは、比較的シンプルで、ユーザーと必要なリソースの点で中規模です。2 番目のシナリオは、それよりも少し複雑になります。最後のシナリオは、データとメタデータの両方のレポーティング要件の点で 最も複雑で、「どのように機能させるか」という点に重点を置きます。ビジネス要件を満たすことができるアーキテクチャーは複数ある可能性がありますが、ここでは 1 つのみを提示します。

以下のプロセスにより、アーキテクチャーとデプロイメント環境を検討します。

1. 図 5-13 (157 ページ) の決定のためのチャートを使用して、適切なデプロイメント・アーキテクチャー (製品構成をサポートするのに必要な環境の数と種類) を決定します。
2. それぞれの製品モジュールをデプロイする環境を決定します。
3. 高可用性の要件を評価します。
4. 各環境のビジネス要件と IT 要件、可用性、および製品構成に基づいてトポロジーを設計します。

Information Server のアーキテクチャーとデプロイメントを評価する前に、各環境のハードウェアで予期される負荷をサポートするのに必要なハードウェア・リソース (CPU コア、RAM、ディスクのボリュームとタイプなど) を判断する必要があります。ただし、サイジングはここでは扱いません。すべての製品モジュールと、それらをサポートするインフラストラクチャーをデプロイする方法を決定した後、各製品モジュールに必要なすべてのリポジトリー資産が各製品モジュールのデプロイ先の環境で確実に使用できるようにするプロセスを確立します。プロセスにおける各シナリオの最終ステップでは、「(実装を) 機能させる方法」を明らかにします。

6.2.1 シナリオ 1: シンプル

中堅規模の製造会社が最近 InfoSphere Information Server を購入して、データ統合のために DataStage と QualityStage の両方を使用する計画を立てています。さらに、同社は Business Glossary も購入しましたが、この時点では全社的に展開する予定はありません。同社は、Metadata Workbench (MWB) も購入して、データ・リネージュ機能のために使用する予定です。

同社は、プロジェクトの初期フェーズでは 3 から 4 名の開発者を配属して、データ統合計画への参加に同意する事業単位が増えた時点で、さらに多くの開発者を追加する予定です。現行の本番要件により、同社は本番環境のプライマリサーバーでの障害に備えて高可用性のフェイルオーバー環境を用意する必要があります。同社は、さまざまな利用可能なオプションを検討した後、シンプルなアクティブ / パッシブ構成が同社のニーズに対応すると判断しました。同社では、開発環境もクラスタリングする利点について議論しましたが、追加のハードウェア・コストの正当性を証明できませんでした。高可用性を考慮して構成される環境は本番環境のみです。

使用するデプロイメント・アーキテクチャーの選択

デプロイメント・ランドスケープを決定するための最初のステップは、決定チャート (図 5-13 (157 ページ)) を確認して適切なデプロイメント・アーキテクチャーを決定することです。このシナリオでは、決定チャートの質問への回答は次のとおりです。

1. DS、QS を使用しているか

   はい

2. IA、BG、MWB、FT を使用しているか

   はい

   重要な決定ポイント : 次の重要な決定ポイントでは、「シンプル」(開発 / テスト / 本番) デプロイメント・アーキテクチャーで十分であるか、あるいはこのシナリオには「ユニファイド」環境に特有の高度な設計機能が必要であるかを判断します。

3. メタデータ・コラボレーションに関連するワークロードには、IA プロファイリングとデータ・ルールまたはメタデータ・レポーティングのための追加のリソースが必要であるか

   いいえ

   説明 : この時点では、Business Glossary の用途が限られており (少数の事業単位に限定)、Metadata Workbench からはデータ・リネージュ・レポートのみが生成されるため、「ユニファイド」開発環境の正当性を証明するのに十分なコラボレーションはないと思われます。また、本番環境では、個別のメタデータ・レポーティング環境が必要となるほどの重大な負荷は生じないと思われます。(本番は、これらの製品モジュールをデプロイするために当然必要な環境です。)

この決定ツリーを使用した結果、開発環境、テスト環境、本番環境で構成される「シンプル」デプロイメント・アーキテクチャーが推奨されます。

製品モジュールのデプロイ先

製品モジュールは以下の環境にデプロイできます。

- DataStage および QualityStage は、標準のソフトウェア開発ライフサイクル (SDLC) 方法論に従ってすべての環境にデプロイされます。
- Business Glossary は、許可ユーザーがアクセスできるように本番環境にのみデプロイされます。グロッサリーの作成作業は組み込みワークフロー・メカニズムによって処理され、無許可ユーザーが承認前のコンテンツを表示するのを防止します。
- Metadata Workbench は確実に本番環境にはデプロイされますが、Metadata Workbench を開発環境にもデプロイする正当な理由が生じる可能性があります。このデプロイメントを決定するには、開発者の要件に関するさらに多くの情報が必要です。

可用性の要件

この例で取り上げている製造会社のビジネス要件によると、同社には、本番環境のみでアクティブ / パッシブの高可用性が必要です。その他すべての問題は同等であり、高可用性構成の簡素化を考慮すると、本番環境では、3 つのサーバー層をすべて同じサーバー (物理または仮想) にインストールすることが推奨されます。

図 6-1 は、3 つの Information Server 層 (サービス、リポジトリー、エンジン) をすべてホストする 1 台のサーバーをベースとした、「シンプル」デプロイメントを示しています。この具体的な構成は、このシナリオの中堅規模製造会社のユース・ケースをサポートする上で開発とテストの両方の環境に適しています。

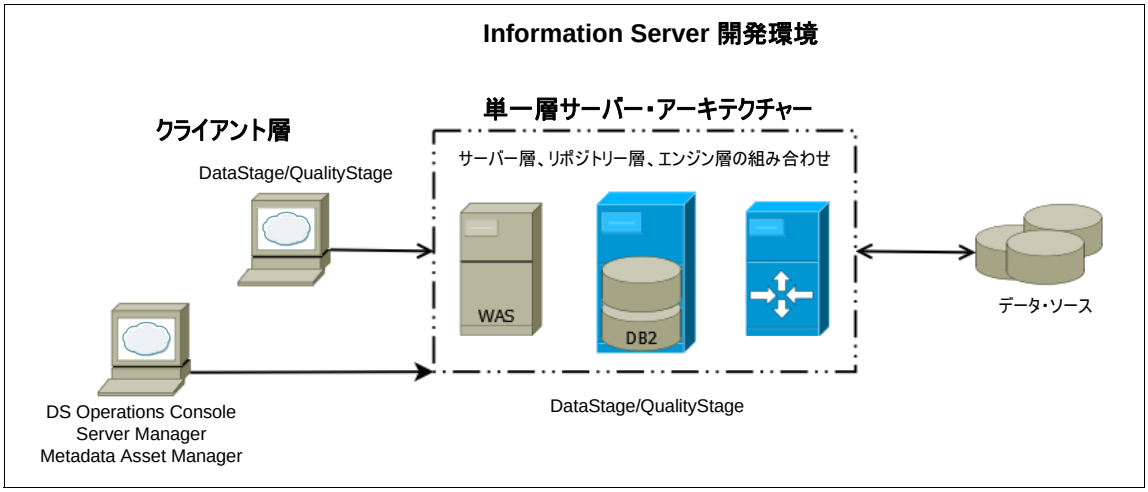


図 6-1　開発環境とテスト環境の「シンプル」デプロイメント・アーキテクチャー

本番環境には、宛先への持続的なデータ・フローを確保するために、アクティブ / パッシブの形式の高可用性が必要です。図 6-2 (163 ページ) は、アクティブ・モードと、2 次サーバーへのフェイルオーバー後の両方の構成を示しています。フロントエンド HA マネージャーがアクティブ・サーバーの「正常性」をモニターしていて、フェイルオーバーが必要となる状況を判別することに注意してください。HA マネージャーは、この目的のために設計されたハードウェア・デバイス、またはパッシブ・サーバーへのフェイルオーバーの必要性を示す主要なプロセスとイベントをモニターしてフェイルオーバー・プロセスを実行するために開発されたソフトウェアまたはスクリプトによって実装できます。アクティブ / パッシブの可用性のアーキテクチャーについて詳しくは、2.7 『Information Server クラスター・インストール』（24 ページ）を参照してください。

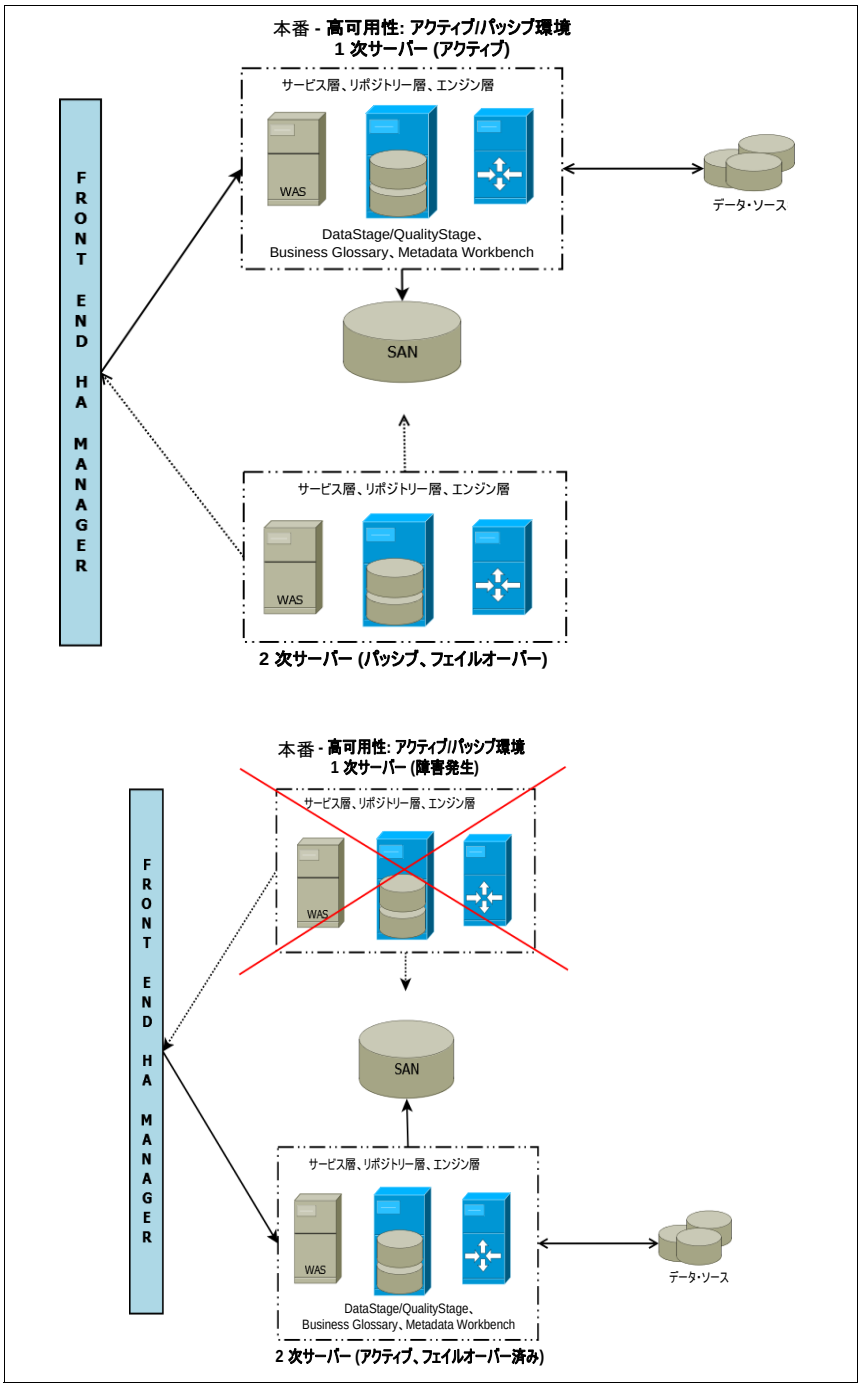


図 6-2　本番環境の「シンプル」高可用性構成

実現方法

63 ページの第 4 章、『Information Server 資産のライフサイクル管理』では、SDLC デプロイメント・プロセスを詳細に説明しています。開発からテスト、そして本番への各環境を通じて DataStage コードをデプロイできる方法を説明しています。DataStage ジョブを本番環境にデプロイするのは SDLC プロセスの一部であるため、必要に応じて、すべての DataStage ジョブデザインメタデータを本番環境の BG と MWB の両方で使用できることが想定されます。そのため、この作業はこのプロセスに含まれません。

このシナリオでは、Business Glossary は、独自のメタデータを作成して、それを BG がデプロイされている本番環境のリポジトリーで保持します。さらに、BG は、本番環境のリポジトリーで既に保持されている他のメタデータ資産 (例えば、本番環境で他の製品モジュールによって使用されるデータベースやデータ・ファイルなどの共通のテクニカル・メタデータ資産) も使用します。BG には、他の環境だけに存在する資産は必要ないため、BG のために他の環境から本番環境に他の資産をデプロイする必要はありません。

BG が使用できる外部のテクニカル・メタデータがいくつかある可能性があります。これを識別して、リポジトリーにインポートし、必要に応じて BG の用語に割り当てる必要があります。BG に役立つ外部のテクニカル・メタデータの代表例は、ビジネス・インテリジェンス・メタデータです。ただし、この場合は、外部メタデータのインポートはデプロイメント・プロセスの一部ではなく、Information Server 製品モジュールの一般的な使用法の一部です。また、一般的に推奨されるステップでは、外部のテクニカル・メタデータを 1 つの環境から別の環境にデプロイするのではなく、必要とされる各環境にインポートします。ただし、この例では、外部のビジネス・インテリジェンス・メタデータは、関連する場合、本番環境にのみ (BG および MWB で使用するために) インポートされます。

データ・リネージュに関するビジネス要件は、Metadata Workbench にリポジトリー内のデータのソースとターゲット (ビジネス・インテリジェンス・メタデータと DataStage ジョブデザインなど) が必要であることを意味します。DataStage ジョブデザインは DataStage のデプロイメント・プロセスの一環としてデプロイされるため、必要なジョブは既に Metadata Workbench で使用できるようにリポジトリーに存在していることが想定されます。ただし、いくつかの小技やコツを利用することで、データ・リネージュを確実にし、場合によっては必要となる手作業を最小限に抑えることができます。

適切なデータ・リネージュ・レポートを生成するための基本的な要件、前提条件、および推奨される手法について詳しくは、「*IBM InfoSphere Information Server によるメタデータ管理*」(SG88-4071) で説明されています。言及すべき重要な 1 つの点は、DataStage ジョブデザインではステージ・パラメーター値としてジョブ・パラメーターを頻繁に使用しますが、Information Server はその値を実行時まで「認識」しないということです。Metadata Workbench はリネージュ・レポートの生成において DataStage ジョブデザインに依存するため、ジョブデザインで変数を使用することは、Metadata Workbench でリネージュ・レポートに必要な重要情報が欠落する可能性があることを意味します。1 つの解決策は、DataStage ジョブの実行結果からオペレーショナル・メタデータ (OMD) を生成、収集、およびロードすることです。この方法により、変数値はインスタンス化されるため、Metadata Workbench で使用して DataStage ジョブデザインメタデータを補足することができます。その他の解決策は、メタデータ資産のデプロイメントに関連していないため、ここでは説明しません。

オペレーショナル・メタデータは、DataStage ランタイム・ファイル・システムで XML 構造化ファイルに収集されます。図 6-3 を参照してください。シェル・スクリプト (図 6-4 (166 ページ)) がオペレーショナル・メタデータ・ファイルを読み取り、そのコンテンツを Information Server リポジトリーにオペレーショナル・メタデータとしてロードすることにより、いくつかのアイテムの中でもパラメーター化された値をインスタンス化します。Metadata Workbench は本番環境にデプロイされ、本番環境で DataStage ジョブを実行することによってオペ

レーショナル・メタデータが生成されるため、このシナリオではオペレーショナル・メタデータを別の環境から本番環境にデプロイする必要はありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<Run StartedAt="2012-07-23T16:45:25" CreationModel="DataStage8" Message="Fi
<Deployment>
<SoftwareResourceLocator>
<LocatorComponent Class="Computer" Name="orion" />
<LocatorComponent Class="SoftwareProduct" Name="DataStage" />
<LocatorComponent Class="SoftwareGroup" SubClass="Project" Name="zeta" />
<LocatorComponent Class="SoftwareExecutable" SubClass="Job" Name="JobDailyR
</SoftwareResourceLocator>
</Deployment>
<Design>
<SoftwareResourceLocator>
<LocatorComponent Class="Computer" Name="orion" />
<LocatorComponent Class="SoftwareProduct" Name="DataStage" />
<LocatorComponent Class="SoftwareGroup" SubClass="Project" Name="zeta" />
<LocatorComponent Class="SoftwareDesign" SubClass="Job" Name="JobDailyRunIn
</SoftwareResourceLocator>
</Design>
<ActualParameters>
</ActualParameters>
<Events>
<Event Type="Read" StartedAt="2012-07-23T16:45:27" FinishedAt="2012-07-23Tl
<DataResourceLocator>
<LocatorComponent Class="Computer" Name="orion" />
<LocatorComponent Class="SoftwareProduct" Name="DB2" />
<LocatorComponent Class="DataStore" SubClass="Database" Name="SAMPLE" />
<LocatorComponent Class="DataSchema" SubClass="Schema" Name="TST_CTRL" />
<LocatorComponent Class="DataCollection" SubClass="Table" Name="POL" />
</DataResourceLocator>
<SoftwareResourceLocator>
<LocatorComponent Class="Computer" Name="orion" />
<LocatorComponent Class="SoftwareProduct" Name="DataStage" />
<LocatorComponent Class="SoftwareGroup" SubClass="Project" Name="zeta" />
```

図 6-3　サンプルのオペレーショナル・メタデータ (OMD) の XML ファイル

```
#!/bin/sh
case "$0" in
  */*) cmd="$0";;
  *) cmd=`which "$0"`;;
esac
dir=`dirname "$cmd"`
nodebin="$dir/."
. "$nodebin/setupEnv.sh"
CLASSPATH=""
CLASSPATH="${CLASSPATH}.:"
CLASSPATH="${CLASSPATH}../conf:"
CLASSPATH="${CLASSPATH}../lib/java/xmeta_cache.jar:"
CLASSPATH="${CLASSPATH}../lib/java/framework.jar:"
CLASSPATH="${CLASSPATH}../lib/java/repository.jar:"
CLASSPATH="${CLASSPATH}../lib/java/log4j-1.2.8.jar:"
CLASSPATH="${CLASSPATH}../lib/java/metadata_services-client.jar
CLASSPATH="${CLASSPATH}../lib/java/metadata_services-asb.jar:"
CLASSPATH="${CLASSPATH}../lib/java/OMDModel_api_gen_ecore.jar:"
CLASSPATH="${CLASSPATH}../lib/java/models_xmeta_api.jar:"
CLASSPATH="${CLASSPATH}../lib/java/ecore-2.2.2.jar:"
CLASSPATH="${CLASSPATH}../lib/java/common-2.2.2.jar:"
CLASSPATH="${CLASSPATH}../lib/java/ecore.change-2.2.2.jar:"
CLASSPATH="${CLASSPATH}../lib/java/ecore.xmi-2.2.2.jar:"
CLASSPATH="${CLASSPATH}../lib/java/ecore.sdo-2.2.2.jar:"
CLASSPATH="${CLASSPATH}../lib/java/commonj.sdo-2.2.2.jar:"
CLASSPATH="${CLASSPATH}../lib/java/runimporter.jar:"
CLASSPATH="${CLASSPATH}${ISF_CP}:${J2EE_CP}"
../apps/jre/bin/java ${J2EE_OPTS} -classpath "${CLASSPATH}" com
```

図 6-4　OMD を IS リポジトリーにロードするための RunImportStart.sh シェル・スクリプト

6.2.2 シナリオ 2: 混合

多数の IBM Information Management 製品を使用している大手銀行が、長年にわたって DataStage を使用しています。この銀行では、最近、別の大規模な銀行と合併するまで、どの環境でも高可用性は必要ありませんでした。合併された開発スタッフは、100 名を超える開発者で構成され、主にバッチ・タイプの ETL 処理を行います。

新しい計画では、Information Analyzer (IA)、FastTrack (FT)、DataStage (DS)、および Metadata Workbench (MWB) を含む Information Server のすべての製品モジュール (Business Glossary を除く) を使用します。

本番環境は、現行の構成からあまり変更されませんが、例外として午前 7:00 にデータをレポーティングのために確実に使用できるようにする高可用性が追加されます。協調的な「ユニファイド」開発環境により、多数の同時ユーザーのために高可用性を実現する必要があります。

使用するデプロイメント・アーキテクチャーの選択

決定チャート (図 5-13 (157 ページ)) を使用して、以下に対する回答を判別します。

1. DS および QS を使用しているか

   はい

2. IA、BG、MWB、FT を使用しているか

   はい (IA、MWB、FT)

3. メタデータ・コラボレーションに関連するワークロードには、IA プロファイリングとデータ・ルールまたはメタデータ・レポーティングのための追加のリソースが必要であるか

   はい (主に FT、および IA とのメタデータ・コラボレーション)

   説明 : この性質のコラボレーションが要件となった時点で、このコラボレーションは、これらの製品モジュール (IA と特に FT) を、それらのメタデータを DataStage 開発者が使用できる開発環境にデプロイする必要があることを意味します。ただし、IA は、(開発環境にデプロイされている場合でも) 本番レベルのデータに対して実行する必要があります。137 ページの第 5 章、『メタデータのデプロイメント・アーキテクチャー』で説明しているように、DataStage の開発作業と IA の作業のために別々のエンジンを構成することによって、この問題を解決できます。DataStage のプロジェクトは 1 つのエンジンで実行され、同時に IA のプロファイリング、ルール実行、または両方が 2 台目のエンジンで実行され、交差することはありません。

   ただし、メタデータ・レポーティング用の別個の環境も必要であるかどうかという問題が残ります。最後の質問に対する回答が答えを決定します。

4. 多数のユーザーがメタデータ・レポートおよび照会を実行し、複雑なグロッサリーがあるか

   いいえ (開発者は多数であるが、FT または IA、およびこれらの製品モジュールから生成されるメタデータ・レポートのユーザーは多数ではない)

製品モジュールの組み合わせがメタデータを共有していても、別々の (開発と本番の) データを処理している状況では、ユニファイド環境が必要になります。ただし、DataStage で ETL 開発作業も行われているため、この銀行の開発標準に適合するテスト環境と本番環境も存在します。メタデータ・レポートに負荷をかける (リネージュ、データ・ルールの結果と照会) ユーザーは少数であるため、現時点では、追加のレポーティングまたはメタデータ専用のレポーティング環境を確立する必要はありません。

製品モジュールのデプロイ先

製品モジュールは以下の環境にデプロイできます。

- DataStage および QualityStage は、標準のソフトウェア開発ライフサイクル方法論に従ってすべての環境にデプロイされます。
- FT マッピング仕様は開発者の支援のためにのみ必要であるため、FT 製品モジュールはユニファイド環境にのみデプロイされます。FT メタデータが本番環境でリネージュのために必要となった場合は、**istool** コマンド・ライン・インターフェース (CLI) の資産交換機能を使用することにより、FT 資産をユニファイド環境からエクスポートして本番 (またはレポーティング) 環境にインポートできます。ただし、現時点ではこのステップは必要ありません。
- IA 製品モジュールは、ユニファイド環境にのみデプロイされ、DataStage の開発者が使用するエンジンとは別のエンジンを使用します。この方法では、DataStage の開発者は開発データのみにアクセスでき、Information Analyzer は本番データのみにアクセスできます。
- コラボレーションが中核となるビジネス要件であるため、MWB もユニファイド環境にのみデプロイされます。MWB で必要となるほぼすべてのメタデータがユニファイド環境で保持されるためです。また、メタデータ専用のレポーティング環境を確立するのに十分な正当な理由はありませんが、リネージュ、ユーザー影響、照会レポートにより、本番環境の負荷が相当に大きくなる可能性が考えられます。ユニファイド環境のトポロジーは、ビジネス要件で規定されているようにこれらの負荷を効果的に処理します。

可用性の要件

ビジネス要件は、ユニファイド環境と本番環境の両方に (本番のシミュレーションのためのテスト環境にも) 高可用性が必要があるとしています。「シンプル」の例 (6.2.1『シナリオ 1: シンプル』（161 ページ）を参照) の本番システムについて推奨される高可用性トポロジーは、この銀行のテスト環境と本番環境にも適しています (図 6-2 (163 ページ) を参照)。

大きな違いはユニファイド環境に関連しています。多数の開発者と Foundation Tools (特に MWB) をサポートするには、サービス層で WebSphere Application Server をクラスタリングすることが推奨されます。このアプローチにより、アクティブ / アクティブに相当する高可用性が実現しますが、さらに重要な点として、多数の同時セッションを Web アプリケーション (つまり、MWB) と共にサポートできます。また、ユニファイド環境には、DataStage の開発ジョブ実行と IA のプロファイリング、ルール実行、または両方をサポートするために別々の 2 つのエンジンが必要になります。

図 6-5 は、このトポロジーのユニファイド・アーキテクチャーのダイアグラムを示しています。

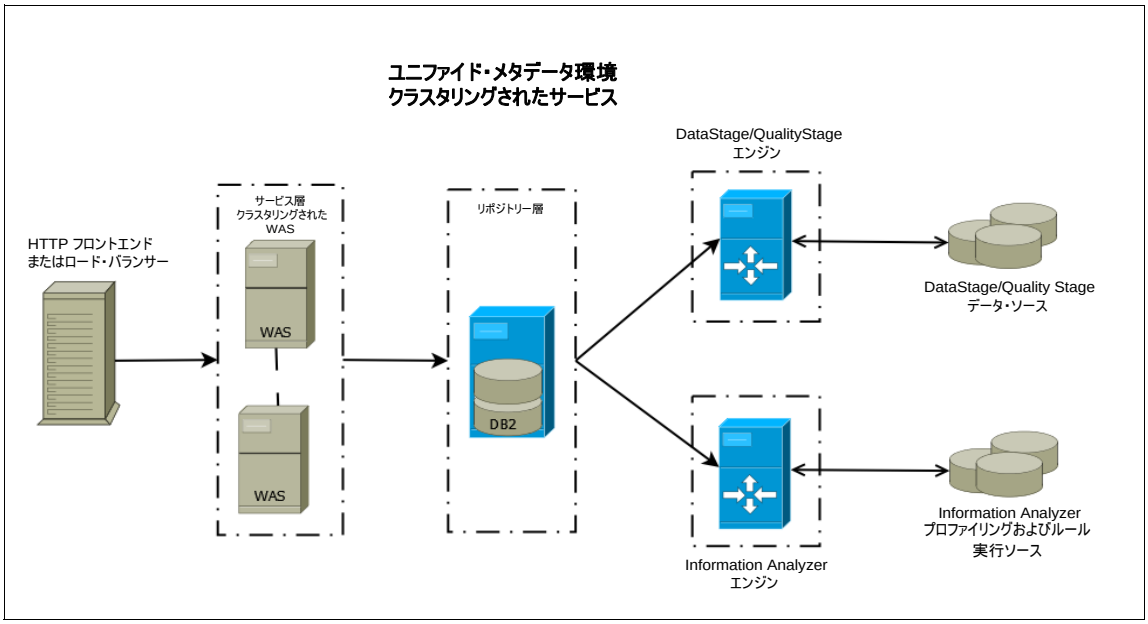


図 6-5　サービス層がクラスタリングされたユニファイド環境

HTTP Web サーバー・フロントエンドと、場合によってはロード・バランサーも、別個のサーバーにインストールするか、WebSphere Application Server クラスター・ノードのいずかと結合することができる点に注意してください。

実現方法

この銀行のデプロイメント・アーキテクチャーは、極めて標準的なものです。例外は、Information Server に固有のユニファイド環境がある点で、InfoSphere Foundation Tools (IA、FT、および MWB など) が組み込まれています。ただし、コラボレーションに関与するすべての Information Server 製品モジュールがユニファイド環境にデプロイされるため、必要なすべての資産を共有リポジトリーで使用できることが必要です。これらは、デプロイされている製品モジュール自体で保持されるためです。

保持されているデータ：Information Server リポジトリーには、企業のデータ・ソースからのいずれのデータも保持されません。これらのソースからのメタデータのみが保持されます。そのため、さまざまなレベルのデータにアクセスする製品モジュール間でリポジトリーは共有されていますが (1 つのエンジンが開発レベルのデータにアクセスして、もう 1 つのエンジンが本番レベルのデータにアクセスします)、リポジトリーで保持されるのは、さまざまなデータ・ソースからのメタデータのみであり、アクセスは適切なユーザーのみに制限されています。

ユニファイド環境に存在するテクニカル・メタデータには、現行の本番資産（元々は開発環境で開発されたため）と、(現在開発中の) 新規開発資産の両方が含まれます。この本番レベルのメタデータにより、Information Analyzer が現行の本番データ・ソース (およびそれらのメタデータ) にアクセスしてそれらを処理するための接続が可能になります。また、現在は本番環境で実行されている DataStage ジョブは、最初は開発 (ユニファイド) 環境で開発されました。したがって、これらのジョブデザインは、ユニファイド環境にも存在していて、Metadata Workbench でリネージュおよび影響分析レポートのために使用できます。

ここで重要な点は、DataStage (および DataStage 開発者) による開発専用データへのアクセスと、IA (および IA ユーザー) による本番レベルのデータのみへのアクセスを完全に分離することです。

開発データと本番データの分離は、以下のステップを使用して、多くの企業の標準的な手法と国際規制に従って実行できます。

1. DataStage の DataStage プロジェクトと IA をサーバー (エンジン層) 上の別々のディレクトリーで作成して、それぞれ独自のエンジンを割り当てます。
2. DataStage からのパラレルジョブは、DataStage エンジンでのみ実行します。
3. IA からのパラレルジョブは、IA エンジンでのみ実行します。
4. DataStage ディレクトリーへのアクセスを適切な DataStage ユーザーにのみ許可します。
5. IA ディレクトリーへのアクセスを適切な IA ユーザーにのみ許可します。

一方で、FastTrack マッピング仕様は、まだ本番環境に配置されていないメタデータを使用する可能性があります。これらの設計仕様は、新しい DataStage ジョブ開発の基礎となります。そのため、このテクニカル・メタデータは開発レベルになります (本番環境で使用可能になるまで)。最後に、開発中の実際のジョブは開発レベルのデータ (および対応する開発レベルのメタデータ) に対して実行される必要があります。

開発レベルと本番レベルのメタデータはすべて、リポジトリーで適切に保持され、それらを使用できる製品モジュールからアクセス可能です。この段階で欠落している唯一のアイテムは、本番環境で DataStage および QualityStage のジョブにより、Metadata Workbench でリネージュと影響分析のために使用されるように生成されるオペレーショナル・メタデータです。このメタデータは、本番環境 *ではなく*、ユニファイド環境のリポジトリーにロードされる必要があります。

6.2.1『シナリオ 1: シンプル』（161 ページ）で説明しているように、オペレーショナル・メタデータは、本番環境で実行されている DataStage および QualityStage のジョブによって DataStage および QualityStage ファイル・システム上の XML ファイルに書き込まれます。そのため、シェル・スクリプトを実行してオペレーショナル・メタデータを本番リポジトリーにロードするのではなく、考えられる方法として、最初に、オペレーショナル・メタデータを含む XML ファイルを本番ファイル・システムから、ユニファイド環境からアクセスできるファイル・システムにコピーまたは移動します。次に、ユニファイド環

境からシェル・スクリプトを実行して、オペレーショナル・メタデータ・ファイルをユニファイド環境のリポジトリーにロードします。

関連 : このステップでは、本番環境で実行されていた DataStage および QualityStage ジョブによって生成されたオペレーショナル・メタデータを関連付けて、それらをユニファイド / 開発環境で保持されている DataStage および QualityStage ジョブデザインに付加します。

6.2.3 シナリオ 3: コンプレックス

大規模な多国籍保険会社が、Solvency II Directive の導入を背景に、企業経営者とすべての事業部門の支持を得て新しい情報ガバナンス・イニシアチブを開始しています。同社は、新しい各ビジネス・プロジェクトで従うべきプロセスを詳細に文書化していますが、ガバナンスの分野における経験はあまりありません。そのため、同社は、このイニシアチブを管理するために Information Server の Enterprise Edition 一式 (およびデータ・セキュリティーのための IBM InfoSphere Guardium® と関連する IBM ソフトウェア) を購入しました。同社は、IBM Cognos によるビジネス・インテリジェンス・レポーティングをサポートするために、新しいデータウェアハウスの設計と開発を開始する予定です。開始にあたり、同社は、保険業界向けの IBM Insurance Information Warehouse (IIW) 業界モデルを組み込みました。InfoSphere Data Architect (DA) モデリング・ツールでニーズに合わせてカスタマイズすることを計画しています。この IBM モデルには、Business Glossary が事前に取り込まれています。同社は、これをリファレンスとして使用して独自のグロッサリーを徐々にビルドし、段階的に全社的に展開する予定です。同社の企業の語彙を標準化するほか、Business Glossary を使用して新しい各プロジェクトのビジネス要件を収集し、可能な限り企業のグロッサリーに対してこれらの要件を標準化します。グロッサリーは、同社のビジネス・メトリック、重要業績評価指標 (KPI)、および一般的な語彙に関連しています。

同社は現在、ETL 設計に使用するソースからターゲットへのマッピング仕様を文書化するためにスプレッドシートを使用しています。このメカニズムに関して同社が直面している問題は、いつビジネス・アナリストが文書を承認したか、他の現行プロジェクトとの要件の重複がある場合に改訂が必要であるかどうか、属性が既に別のスプレッドシートにマップされているかどうかを認識できないことです。これらの問題はいずれも、貴重なリソースを無駄にして、ミスが発生しやすくなる原因となっています。この保険会社は、スプレッドシートの代わりに FastTrack を使用して、これらのマッピング仕様を集中管理することを計画しています。これには、既存のデータ・ソースを再使用するための表示、検索、および「発見」のためのメカニズムも含まれます。最初に同社は、Netezza® ベースのデータウェアハウスで設計が完成およびインスタンス化されるまで、ETL 設計のターゲットとして物理設計モデル (DA の) を使用します。次に、同社は、Netezza プラットフォーム上に実装されたデータウェアハウスのメタデータへの再マップを自動的に行います。

ETL の観点から見ると、同社は、既存のデータウェアハウスを新規モデルにマイグレーションして、既存のソースから新規データウェアハウスへの現行の ETL プロセスのリエンジニアリングを行う必要があります。残念ながら、同社の現行の ETL プロセスをサポートするために使用できる文書は、(コード自体を除くと) マッピング・スプレッドシートだけです。これは、この目的には適していません。新しいイニシアチブの一環として、同社は、メタデータを IS 共有リポジトリーにインポートすることにより、文書化の目的でデータ・ソースをカタログに登録します。開発者は、実施するリエンジニアリング作業の影響を理解する必要もあるため、影響分析機能のために Metadata Workbench を使用します。

同社は、現在は別の ETL ツールによって実装している新しい ETL プロセスのすべてにわたって幅広く DataStage を使用する予定です。また、DataStage は、データ・ソースからの実際のメタデータへの参照として、共有リポジトリーの

データ・カタログを使用できます。このカタログは、DataStage ジョブデザインで使用される DataStage 表定義で使用できます。企業全体でタイム・ゾーンが幅広く、時間の制限枠が比較的短いため、毎日、比較的短い時間フレームの中でいくつかの個別のプロジェクトと大量のデータが処理されます。

ガバナンス・イニシアチブの一環として、この保険会社は、データの品質が同社のデータ品質メトリックを満たしていることを検証する必要があります。同社は、品質ルールを定義して、これらの定義された標準の順守を定期的に測定します。さらなる品質の観点から見ると、同社は、Solvency II のための規制報告書の基となるデータが Solvency II 報告書として適切であることを検証する必要があります。監査員は、Metadata Workbench を使用して、規制報告書を監査証跡に含めるためにそのデータ・リネージュを検査します。

使用するデプロイメント・アーキテクチャーの選択

決定チャート (図 5-13 (157 ページ)) を使用して、以下に対する回答を判別します。

1. DS または QS (または ISD) を使用しているか

   はい

2. IA、BG、MWB、FT を使用しているか

   はい (上記のすべて)

3. メタデータ・コラボレーションに関連するワークロードには、IA プロファイリングとデータ・ルールまたはメタデータ・レポーティングのための追加のリソースが必要であるか

   はい (相当量のコラボレーションとレポーティング)

4. 多数のユーザーがメタデータ・レポートおよび照会を実行し、複雑なグロッサリーがあるか

   はい (定期的な品質メトリック・レポート、メタデータ影響分析、およびデータ・リネージュ・レポートのほか、完全な参照用グロッサリーと、割り当て済み資産と他の関係を含む徹底的にビルドされたガバナンス対象グロッサリーが企業で展開されるため、時間とともにユーザー数は増加する)

   説明 : リポジトリーには、非常に大量のメタデータが、用語とさまざまな技術資産 (データ・ソース、ターゲット、BI 資産) の間の多数の関係と共に保持されます。(RDBMS は一般にクラスタリングを必要とすることなく大量のデータをサポートできますが) このように保持されるということは、サービス層に、場合によってはリポジトリー層にも大きな負荷がかかることを意味します。メタデータ・レポーティングは、本番環境またはユニファイド環境のいずれかに対して過度の影響を与える可能性があります。

コラボレーションのレベル (DA および FT) とメタデータ・レポーティング要件 (IA データ・ルール・メトリック、Business Glossary のガバナンス、Metadata Workbench の影響分析とデータ・リネージュ) の両方を考慮すると、この保険会社は、DataStage SDLC のための開発、テスト、本番の環境と、メタデータ・レポートおよびデータ品質メトリックのための別のレポーティング環境を確立する必要があります。コラボレーションのために開発環境をユニファイド環境にできる可能性はあります。ただし、データのプロファイリングのために IA が使用されていないため、関連する製品モジュールを複数の環境にデプロイするのではなく、協調的な資産をレポーティング環境で作成して (必要な場合は) 開発環境にデプロイすることができます。

また、FT とのコラボレーションは、ユニファイド環境の必要性も示唆しています。しかし、最適な解決策は、FT を Business Glossary と一緒にデプロイすることです。また、実際のマッピング仕様を DataStage と同じ環境に配置する必要は

ありません (FT が DataStage テンプレート・ジョブの生成に使用される場合は除きますが、このユース・ケースとは異なります)。

このお客様 (またはすべてのお客様) に最適なデプロイメント・アーキテクチャーを実現するための解決策は複数ある可能性があります。ただし、デプロイメント・アーキテクチャーを決定する際の中核となる原則の 1 つは、可能な限りシンプルにすることです。この保険会社の場合は、製品モジュールをデプロイする環境の数を最小限に抑えることは有用です。1 つの環境から別の環境へのリポジトリー資産 (ビジネス・メタデータとテクニカル・メタデータ) のデプロイメントが最小限に抑えられるためです。また、ETL 処理のための 開発 / テスト / 本番とメタデータ・レポーティングのためのレポーティングという 2 つの異なるデプロイメント・アーキテクチャーも必要です。その後、これらの 2 つのデプロイメント・アーキテクチャーの間で共有する必要がある成果物を判別できます。

製品モジュールのデプロイ先

製品モジュールは以下の環境にデプロイできます。

- DataStage (および必要な場合は QualityStage) は、標準のソフトウェア開発ライフサイクル (SDLC) 方法論に従って 3 つの標準環境 (開発、テスト、本番) にデプロイされます。また、データ・リネージュ・レポートを形成する際に MWB で使用される DataStage ジョブをインポートするために、これらをレポーティング環境にもインストールする必要があります。
- Information Analyzer は、データ・ルール実行と、(本番レベルのデータに関する) これらのメトリックのデータ品質レポーティングのために使用されます。そのため、これはレポーティング環境に属します。この機能は、コラボレーションの目的には使用されず、レポーティングの目的にのみ使用されるためです。

  次の状況を検討します。DataStage 開発者も DataStage データ・ルール・ステージでこれらのデータ・ルールを使用する場合は、コラボレーションが存在し、レポーティング環境に加えて、またはレポーティング環境の代わりに開発環境に IA をデプロイする必要があります。同様に、IA がプロファイリングのために使用されている場合は、これはもう 1 つのコラボレーション機能であるため、開発 (ユニファイド) 環境でのデプロイメントの正当な理由になります。ただし、この状況は、この保険会社には該当しません。IA はレポーティング環境にのみデプロイされます。
- Business Glossary には独自の組み込みワークフローがあります。つまり、1 つの環境にのみデプロイする必要があります。この状況下では、Business Glossary はレポーティング環境にデプロイされます。その主目的は、必要に応じたユーザーによるグロッサリーの検索、表示、および照会であるためです。ただし、このアプローチは、Business Glossary が関連付ける必要があるすべてのメタデータがレポーティング環境で保持されている (またはコピーが保持されている) 必要があることを意味します。
- FastTrack はコラボレーションの目的で開発 (またはユニファイド) 環境を拠点としますが、最適なデプロイ先は Business Glossary と同じ環境です。これにより、ソースからターゲットへのマッピングの「発見」を「自動化」するために用語とデータベース列の関係を使用できます。したがって、FT はレポーティング環境にデプロイされます (必要な場合は、メタデータを開発環境にデプロイするオプションがあります)。
- Metadata Workbench は、データ・リネージュ・ガバナンス・レポートと、開発者による影響分析の両方に使用されます。そのため、Metadata Workbench は、開発環境（まだ本番に配置されていない新規ソースの影響分析のため）と、レポーティング環境（本番資産に対するデータ・リネージュのため）の両方にデプロイされる必要があります。代替手段として Metadata Workbench をレポーティング環境にのみデプロイすることは可能ですが、その場合は、開発環境からレポーティング環境へ、非常に多くの技術資産と、レポーティング環境の他の製品モジュールのいずれにも必要でない技術資産をデプロイする必要が生じます。

要約すると、DataStage および QualityStage は、開発、テスト、本番環境にデプロイされ、またレポーティング環境にもデプロイされます。その他のすべての製品モジュールはレポーティング環境にのみデプロイされます。

可用性の要件

デプロイメントは、次の 2 つの異なるアーキテクチャーに分かれます。

- DataStage の開発 / テスト / 本番
- Foundation Tools (IA、FT、BG、MWB) のレポーティング

したがって、可用性のメカニズムはデプロイメント・アーキテクチャーごとに固有です。

ビジネス要件は、この保険会社の国際性のために非常に大量のデータが比較的短い時間の制限枠で処理されることを示しています。このような状況下では、グリッド・アーキテクチャーが管理容易性のために最も優れた柔軟性とスケーラビリティーを提供します。

本番環境では、最もシンプルなトポロジーは、1 台のマシンにサービス層およびリポジトリー層をエンジン層のヘッド・ノード (リソース・マネージャーも含む) と一緒に配置することです (図 2-20 (45 ページ) を参照)。必要な計算ノードの数によっては、各計算ノードを別々のマシンに配置できます。処理されるデータの量が時間とともに増える場合、さらに多くのノードを簡単に追加できます。

グリッド・トポロジーのもう 1 つの利点は、ある時間の制限枠 (通常は夜間) には本番環境にサービスを提供する同じ計算ノードが、日中は開発環境とテスト環境にもサービスを提供できるため、ハードウェアの使用率を最大限に高め、他の環境にさらに多くの処理能力を提供して、本番環境のシミュレーションがより正確になることです。

この保険会社の場合は、開発およびテストのトポロジーは本番環境と同一です (ただし、ハードウェア仕様は、CPU コアと RAM の点で異なる可能性はあります)。その理由は、ビジネス要件に従い、トポロジーを複雑化させる正当な理由がないためです。ただし、アクティブ / パッシブの高可用性のために構成するかクラスタリングされている個別のサーバーに各層を配置して、引き続きグリッド (計算ノード) を本番環境と共有できない理由はありません。ただし、グリッドがすべての環境で共有されなければならない理由もありません。図 6-6 (174 ページ) に、3 つの環境がグリッド計算ノードを共有する様子を示します。

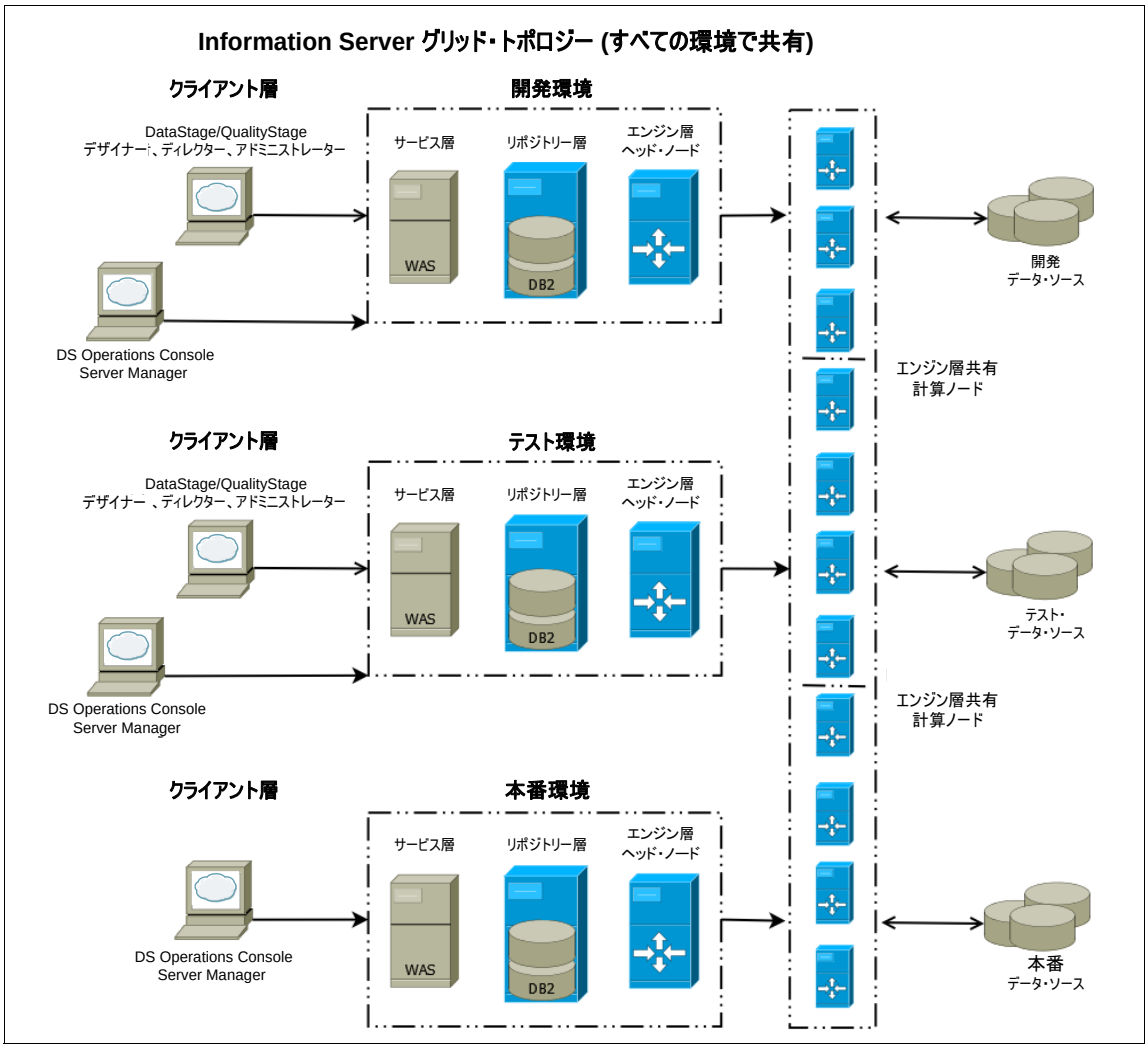


図 6-6　開発、テスト、本番環境で共有されるグリッド・トポロジー

4 つ目の環境はレポーティング環境です。ビジネス要件によると、用語と技術資産の関係と用語間の関係 (特定の企業によって開発された用語と IIW 参照用グロッサリーとの関連性など) が含まれる高度なビジネス・グロッサリーが存在します。また、ユーザーはデータ・リネージュ・レポートに依存するため、場合によってはサービス層での処理が多くなります。その結果、(始めは) 2 つの WebSphere Application Server ノードでクラスタリングされたサービス層のセットアップが推奨されます。

この環境における唯一のパラレルエンジン要件は、IA によるモニターとレポーティングのためのデータ品質ルールです。実行は毎日行われないため、実行時間の制限枠は柔軟に設定できます。そのため、エンジン層をリポジトリー層と一緒に配置でき、汎用操作への干渉はありません。ただし、将来、エンジン上の負荷がパフォーマンスに悪影響を与える場合は、このエンジンがグリッドとも計算ノードを共有する可能性があります。

レポーティング環境のトポロジーは、シナリオ 2 のユニファイド環境に似ています。例外は、シナリオ 3 のエンジンが 1 つだけであることです。図 6-7 に、レポーティング環境のトポロジーの外観を示します。

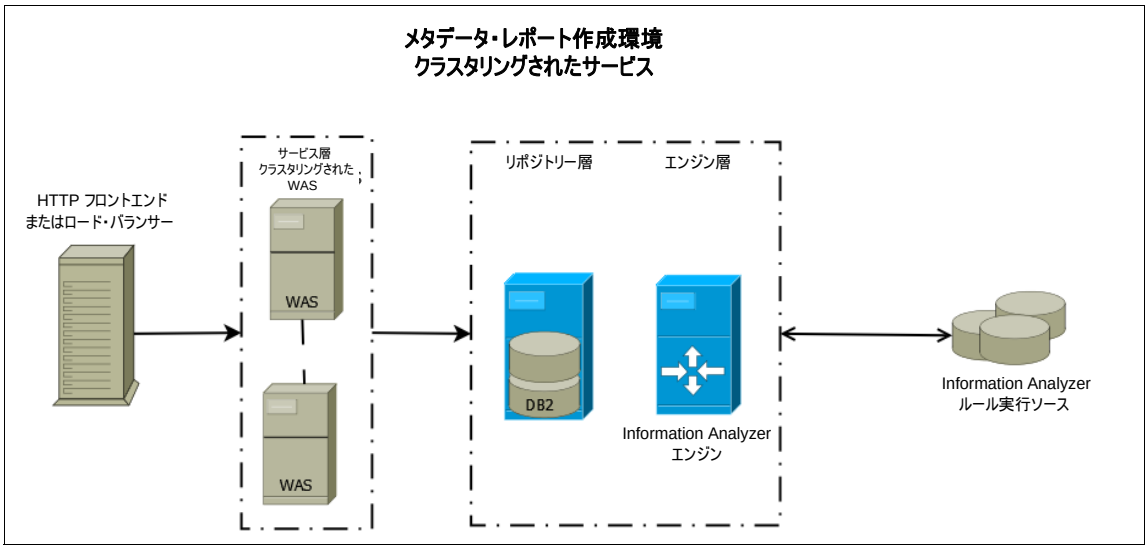


図 6-7　WebSphere Application Server (サービス層) がクラスタリングされたメタデータ・レポーティング環境

機能させる方法

繰り返しになりますが、63 ページの第 4 章、『Information Server 資産のライフサイクル管理』で説明している標準的な SDLC 方法論 (ソース制御ソフトウェアと IS Server Manager を使用して開発からテスト、そして本番に DataStage ジョブをプロモートする) を、この保険会社での DataStage による ETL 処理で使用できます。特に、この方法論は情報ガバナンス・イニシアチブの支援となります。

Metadata Workbench をレポーティングインスタンスと開発インスタンスの両方にデプロイすることが推奨されます。ただし、最初に開発環境を提示します。Metadata Workbench は、DataStage ジョブ実行からデータ・ソースのテクニカル・メタデータ、DataStage ジョブデザイン、およびオペレーショナル・メタデータを使用します。テクニカル・メタデータのインポートには、InfoSphere Metadata Asset Manager または DataStage 自体を使用できます。この環境は開発であるため、メタデータをロードするためのどの方法でも十分です (一貫した方法で保管されている限り)。ジョブデザインは、DataStage が保持されているリポジトリー内に既に存在します。最後に、オペレーショナル・メタデータは、開発環境の実行中のジョブから収集されるため、変更は不要です。Metadata Workbench によって使用されるすべてのメタデータは、開発環境にインポートされるか、開発環境で生成されて保持されるため、すべての Metadata Workbench レポートを使用できます。影響分析レポートとデータ・リネージュ・レポートの生成について詳しくは、下記の Metadata Workbench ユーザー資料を参照してください。

http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r7/topic/com.ibm.swg.im.iis.mdwb.admin.doc/topics/ct_administeringMetadataWorkbench.html

本セクションの残りの部分では、レポーティング環境と、Information Server 製品モジュールによって生成されるメタデータの管理方法に重点を置きます。

以下のステップを使用します。

1. 関連する外部メタデータは、Information Server のレポーティングイングインスタンスにインポートされる必要があります。InfoSphere Metadata Asset Manager および InfoSphere Data Architect のメタブリッジ・ブローカーを使用して、新しいデータウェアハウスの InfoSphere Data Architect の論理データ (LDM) および物理データ (DBM) モデルをリポジトリーにインポートします。InfoSphere Metadata Asset Manager を使用してメタデータをインポートする方法について詳しくは、下記の資料を参照してください。

   http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r7/topic/com.ibm.swg.im.iis.mmi.doc/topics/t_importing_and_sharing.html

2. InfoSphere Metadata Asset Manager コネクターを使用して (可能な場合)、現時点で本番環境で使用されているすべての既存のデータ・ソースからメタデータをインポートします。これにより、ETL 設計段階で使用されるマッピング仕様を作成するために必要な大量のメタデータを得られます。ソース・メタデータと保管されている接続は、IA により、データ品質ルール・メトリックを実行するためにデータ・ソースに接続するプロセスで再使用されます。
3. Business Glossary の管理タブから、IBM Industry Model に付属の IIW グロッサリーをインポートします。これには、グロッサリーの用語と、DA からインポートされた物理モデルとの間の関係が含まれています。このようにモデルの意味体系を強化すると、ソースからターゲットへのマッピング作業を行うデータ・アナリストの支援となります。
4. InfoSphere Metadata Asset Manager Cognos メタブリッジ・ブローカーを使用して、関連するビジネス・インテリジェンス・レポートも Cognos からこのリポジトリーにインポートします。このメタデータは、Business Glossary での用語の割り当てと、Metadata Workbench でのデータ・リネージュ・レポーティングで使用できます。

BI レポート : BI レポートはプロジェクトの後の段階になるまで使用できない可能性がありますが、レポートが使用可能になった時点でメタデータをインポートする必要があります。

DataStage ジョブが本番環境にプロモートされるのと同時に、ジョブもレポーティング環境にプロモートされます。Metadata Workbench は、DataStage ジョブデザインを使用して、データ・リネージュと影響分析の結果を判別できるようにします。この作業は継続的に行われるため、このステップ (IS レポーティングインスタンスへの DataStage コードのプロモート) を標準の SDLC の手順に追加する必要があります。幸い、レポーティングインスタンスで DataStage コードのプロモートが必要となるのは、コードが本番にプロモートされる時点と厳密に同じであるため、このステップは簡単です。

注意 : DataStage ジョブを Information Server のインスタンスにインポートするには、(IA のみが使用する) エンジンがインストールされているだけでは不十分です。そのため、DataStage ジョブが実行されなくても、DataStage をレポーティングインスタンスにインストールすることも必要です。

DataStage の「非実行」インスタンスのインストールに関して特別なライセンス交付の規則が存在します。詳細については、IS 販売担当員にお問い合わせください。

DataStage ジョブが本番環境で実行される際、オペレーショナル・メタデータ (OMD) は*本番*環境から収集されますが、*レポーティング*環境にロードされます。本番環境では他の製品モジュールによって使用されないため、本番環境へのロードは必要*ありません*。本番環境でのオペレーショナル・メタデータの収集およびロードのプロセスについては、6.2.1『シナリオ 1: シンプル』（161 ページ）で説明しています。

シナリオ 3 では、オペレーショナル・メタデータを収集するプロセスは同じままですが、ロード段階ではさらに 2 つのステップがあります。

1. 本番環境から、オペレーショナル・メタデータが保管されている XML ファイルを、レポーティング環境のエンジン層によって使用されているファイル・システムに移動します。
2. XML ファイルを編集します。オペレーショナル・メタデータをリポジトリーにロードするシェル・スクリプトを実行する前に、エンジン層をホストするサーバーの名前を本番サーバーのホスト名からレポーティング・サーバーのホスト名に変更します。

図 6-8 に、変更するエンジン・ホスト名が入っている、オペレーショナル・メタデータ・ファイル内の XML タグを示します。図 6-9 には、レポーティング環境のエンジンを識別するよう変更済みの内容を示します。

```
<LocatorComponent Class="Computer" Name="prodHost" />
<LocatorComponent Class="SoftwareProduct" Name="DataStage" />
```

図 6-8 XML オペレーショナル・メタデータ・ファイル内の実際の本番エンジン識別タグ

```
<LocatorComponent Class="Computer" Name="reportHost" />
<LocatorComponent Class="SoftwareProduct" Name="DataStage" />
```

図 6-9 レポート「エンジン」を識別する変更済み XML タグ値

注意 : XML ファイルでは、以下のタグが頻繁に使用されます。

<LocatorComponent Class="Computer" Name="prodHost" />

ただし、下記のタグと一緒に使用される場合は、ホスト名を <prodHost> から <reportHost> に変更する必要があります。

<LocatorComponent Class="SoftwareProduct" Name="DataStage" />

prodHost の名前のすべてのインスタンスに対して自動的な検索と置換の操作を実行しないでください。

RDBMS (例えば、DB2、Oracle、Teradata) などの他の “SoftwareProduct” には、データ・ソースが置かれているホスト (エンジンが置かれているホストではなく) を識別する ‘“Computer” Name=’ もあります。ただし、‘”SoftwareProduct” Name=”DataStage”’ の場合に限り、ホスト名 ID をレポーティング・エンジンのホスト名に変更する必要があります。

本番エンジンの代わりにレポーティング・エンジンを組み込むためにすべての XML ファイルを変更した後、RunImportStart.sh シェル・スクリプトを使用して、オペレーショナル・メタデータをレポーティングインスタンスのリポジトリーにロードする必要があります。この変更が必要となるのは、Metadata Workbench が、レポーティングインスタンスのエンジン層の DataStage プロジェクトにインポートされた DataStage ジョブデザインにオペレーショナル・メタデータを関連付けられるようにするためです。DataStage ジョブは開発環境で開発され、本番環境で実行されますが、DataStage ジョブをインポートすると、それらはローカル・エンジンに結びつけられるため、オペレーショナル・メタデータを一致させる必要があります。

これで、レポーティングインスタンスには、製品モジュールが処理の実行に必要とするすべての資産が存在することになります。

6.3 まとめ

本書では、Information Server のインフラストラクチャー、さまざまなアーキテクチャー・トポロジー、製品スイートを構成する製品モジュール、それらの機能と基本的な用途、および製品モジュールの統合について説明しました。また、DataStage および QualityStage については、SDLC およびその管理技法についても説明しました。これらの製品は、IS 製品スイートにおいて主要なコード開発製品モジュールであるためです。その一方で、本書では、メタデータが製品モジュール間の統合の主要な部分を形成していることも説明しています。メタデータは単にさまざまな製品モジュールの使用による副産物ではありません。メタデータは、ETL プロセスによって得られるデータと同様に、それ自体で成果物です。したがって、メタデータのライフサイクルは、DataStage および QualityStage の SDLC とは大幅に異なる要素として導入され、また、異なる方法で管理される必要があります。

ビジネス・インテリジェンス、データウェアハウス、マスター・データ管理、およびその他の目的のために (ETL プロセスから) 生成されるデータと、ガバナンス、影響、リネージュ、およびその他の目的のために (製品モジュールの生成とインポートから) 生成されるメタデータの両方を確実に実現できるように、本書では、ビジネス目標を達成する上で適切なデプロイメント・アーキテクチャーを決定するためのデプロイメントの戦略と技法をいくつか紹介しています。これらの戦略では、製品構成、ユーザー群の構成、使用上の動作、レポーティングの要件、可用性の要件、および Information Server インフラストラクチャーのさまざまなコンポーネントに対する製品モジュールごとの負荷とストレスを考慮しています。

本書で提示している Information Server の基本原理と本章の例の具体的な分析を理解することにより、ビジネス目標を達成できる Information Server 製品モジュール、これらの製品モジュールを最適な方法でデプロイして最大限のメリットを得る方法、選択した IS 製品の配置を確実に成功させるアーキテクチャー・トポロジーのタイプを判別できます。

用語集

分析データベース (analysis database) InfoSphere Information Analyzer が分析ジョブの実行時に使用して、広範な分析情報を保管するためのデータベース。
分析データベースは、IADB という用語で呼ばれることもある。分析データベースには、InfoSphere Information Analyzer プロジェクト、分析結果、設計時の情報は含まれない。これらの情報はすべて、Information Server リポジトリーに保管される。

ベースライン分析 (baseline analysis) 列分析の結果の保存されたビューと、後で収集される同じ結果の別のビューを比較するタイプのデータ分析。

ビジネス・アナリスト (business analyst) ビジネス・ニーズと問題を分析して、ユーザーおよびステークホルダーと協議して情報技術によって事業収入を改善できる機会を特定し、要件を技術的形式に変換する専門家。

ブリッジ (bridge) メタデータ・エレメントを標準モデルにマップすることによってメタデータを 1 つの形式から別の形式に変換するコンポーネント。このモデルは、ソース・ツールの意味体系をターゲット・ツールの意味体系に変換する。例えば、ソース・ツールはビジネス・インテリジェンスまたはデータ・モデリングのツールの場合があり、ターゲット・ツールは Information Server リポジトリーの場合がある。あるいは、ソース・ツールが Information Server リポジトリーで、ターゲット・ツールがデータ・モデリング・ツールで使用されるファイルの場合もある。「*コネクター (connector)*」および「*Information Server リポジトリー (Information Server repository)*」も参照。

ビジネス・インテリジェンス資産 (business intelligence assets) データのビジネス・ビューを提供するレポートおよびモデルを編成するためにビジネス・インテリジェンス (BI) ツールによって使用される情報資産。このような資産には、BI レポート、BI モデル、BI コレクション、およびキューブが含まれる。

基数 (cardinality) 列に含まれる固有値の数の指標。

Citrix Systems IBM と協力して、デスクトップ仮想化、サーバー仮想化、ネットワーク最適化、およびコラボレーション製品を提供し、組織がよりシンプルかつ費用対効果に優れた IT 環境を形成できるように支援する企業。

クライアント層 (client tier) InfoSphere Information Server スイートと製品モジュール、およびそれらがインストールされているコンピューターの開発、管理、およびその他の作業に使用されるクライアント・プログラムおよびコンソール。

列分析 (column analysis)F データの状態をフィールド・レベルで記述するデータ品質プロセス。列分析を実行するには、列分析ジョブを定義して実行してから、結果を検討する。

共通ドメイン (common domain) 列は、同じ値を共有するときに共通ドメインを共有する。

計算ノード (compute node) パラレル処理環境でジョブ・ロジックのエレメントを処理する処理ノード。コンダクター・ノードではない処理ノードはすべて計算ノードである。各エンジン層には、1 つ以上の計算ノードを配置できる。「*処理ノード (processing node)*」および「*コンダクター・ノード (conductor node)*」も参照。

コンダクター・ノード (conductor node) ジョブ実行を開始する処理ノード。各エンジン層に、コンダクター・ノードを 1 つのみ配置できる。「*処理ノード (processing node)*」も参照。

コネクター (connector) リレーショナル・データベースやメッセージング・ソフトウェアなどの外部データ・ソースでデータ接続とメタデータ統合を提供するコンポーネント。
コネクターには通常、外部データ・ソースに固有のステージが含まれる。「*ブリッジ (bridge)*」も参照。

定数 (constant) 処理に使用される不変の事前定義値を持つデータ。

クロスドメイン分析 (cross-domain analysis) 2 つのデータ列の間のデータ値の重複を識別するタイプの分析。

クロス表分析 (cross-table analysis) 外部キー分析とクロスドメイン分析を組み合わせたタイプの分析。

データ・アナリスト (data analyst) データ分析プロセスの作成、実行、および検討を行う技術的専門家。

データ・クラス (data class) データ・フィールド内のデータの論理タイプを指定する分類。例えば、INDICATOR 分類は、TRUE/FALSE または YES/NO などのバイナリー値を表す。

データ・フィールド (data field) ファイルまたは表のすべてのレコードに共通する特定のデータ値のセットが含まれる列またはフィールド。

データ・ファイル (data file) (1) フィールドの集合 (データ・ファイル構造と呼ばれる) をデータベース表ではなくネイティブ・システム・ファイルに保管するファイル。
(2) 単一ファイルに保管されているフィールドの集合を表す情報資産。例として、フラット・ファイル (階層構造を持たないファイル)、複合フラット・ファイル (メインフレーム・データ構造と XML ファイルなど、階層構造を持つファイル)、または順次ファイルが挙げられる。

データ・プロファイリング (data profiling) ソース・データの品質と構造を理解すること。

データ・ルール (data rule) データ・プロファイリング時およびデータ品質評価時に検出された条件を評価および分析するデータ・ルール定義から生成された式。

データ・ルール定義 (data rule definition) データ・レコードを評価するためのさまざまな条件を含む True/False (ブール) 式。

データ・セット (data set) パラレルデータ・ファイルと、それらを参照するディスクリプター・ファイルのセット。
データ・セットは、パーティション化の度合いを保存することによってディスクへのデータの書き込みを最適化する。「*データ・ファイル (data file)*」も参照。

データ構造 (data source) データ自体のソース (データベースや XML ファイルなど) と、データにアクセスするために必要な接続情報。

データ・サブクラス (data subclass) データのタイプをさらに具体的に指定するために使用されるユーザー定義のサブクラス。例えば、「Quantity」クラス・カテゴリーの中で、サブクラス 01 は、通貨に関する情報を保持するデータ・フィールドを表すことがある。01 サブクラス・カテゴリーは、このコンテキストでは 1 ドルを意味する。

災害復旧 (disaster recovery) 災害によって 1 次サイト全体が破壊されたり、作動不能になった場合に、データ・センターを別のサイトの別のハードウェアでリカバリーできる能力。

ドメイン (domain) 列の中の一連のデータ。

ドメイン層 (domain layer) 「*サービス層 (services tier)*」を参照。

DS エンジン (DS engine) 「*InfoSphere Information Server エンジン (InfoSphere Information Server engine)*」および「*サーバー・エンジン (server engine)*」を参照。

エンジン (engine) 「*InfoSphere Information Server エンジン (InfoSphere Information Server engine)*」を参照。

エンジン層 (engine tier) InfoSphere Information Server スイートおよび製品モジュールのエンジン・コンポーネント (InfoSphere Information Server エンジン・コンポーネントやサービス・エージェントなど) と、それらのコンポーネントがインストールされている 1 台以上のコンピューターの論理グループ。

外部キー分析 (foreign key analysis) キーと関係の機能の一部となっているタイプの分析。この分析は、表データで外部キー候補を識別することによって、表間の関係を発見する。外部キーは、1 次キー分析中に既に定義または識別されている 1 次キーを参照する。

フォーマット度数 (format frequency) 列で使用される汎用フォーマットの度数分布。

度数分布 (frequency distribution) 列の各固有値の出現回数。

フロントエンド・ノード (front-end node) 「*コンダクター・ノード (conductor node)*」を参照。

汎用フォーマット (general format) 各固有データ値での文字記号の使用。例えば、列の中のすべてのアルファベット文字は文字 A に置換される。

グローバル論理変数 (global logical variable) ファクトまたは特定のデータ部分を表すために設定する値。すべてのデータ・ルール定義およびデータ・ルールで使用できる共有構文である。

ヘッド・ノード (head node) 「*コンダクター・ノード (conductor node)*」を参照。

高可用性 (high availability) ハードウェア障害発生時に素早くリカバリーできるシステム。

推論 (inferences) 可能性が信頼度として解釈される統計的推論。

予想データ・タイプ (inferred data type) 列分析ジョブの実行中に列の値が検査されるとき、値は、さまざまなデータ・タイプに関して検査される。予想データ・タイプのリストから、検討プロセス中に表示するのに最適なデータ・タイプが推奨される。

Information Server リポジトリー (Information Server repository InfoSphere Information Server スイートの製品モジュールの設計時、実行時、グロッサリー、およびその他のメタデータを保管する共有コンポーネント。Information Server リポジトリーは、Information Server リポジトリー層にインストールされる。

Information Server リポジトリー層 (Information Server repository tier) Information Server リポジトリーと、インストールされている場合は InfoSphere Information Analyzer データベース (分析データベース) およびこれらのコンポーネントがインストールされている 1 台以上のコンピューター。

InfoSphere Information Server エンジン (InfoSphere Information Server engine) 発見、分析、クレンジング、または変換などのタスクまたはジョブを実行するソフトウェア。このエンジンには、サーバー・エンジン、パラレルエンジン、および InfoSphere Information Server と製品モジュールのランタイム環境を構成するその他のコンポーネントが含まれる。

InfoSphere Information Server インスタンス (InfoSphere Information Server instance) エンジン層、サービス層、および Information Server リポジトリー層の一部を成すインストール済みコンポーネントのセット。

入力リンク (input link) データ・ソースをステージに接続するリンク。「*リンク (link)*」も参照。

ジョブ (job) データ・ソースに接続して、そのデータを抽出および変換し、その後でデータをターゲット・システムにロードすることができる設計オブジェクトおよびコンパイル済みプログラマチック・エレメント。
ジョブのタイプには、パラレルジョブ、シーケンスジョブ、サーバー・ジョブ、およびメインフレーム・ジョブが含まれる。「*ジョブデザイン (job design)*」および「*ジョブ実行可能ファイル (job executable)*」も参照。

ジョブデザイン (job design) ジョブの中で使用されるソースとターゲットを定義するメタデータと、関連データに作用するロジック。ジョブデザインは、ステージと、これらのステージ間のリンクで構成される。開発者は、ジョブデザインを作成するためにビジュアル・ツールを使用する。ジョブデザインは、ジョブ実行可能ファイルから分離されて Information Server リポジトリーに保管される。「*ジョブ (job)*」、「*ジョブ実行可能ファイル (job executable)*」、「*ステージ (stage)*」、および「*リンク (link)*」も参照。

ジョブ実行可能ファイル (job executable) ジョブの実行時に使用されるバイナリー・オブジェクト、生成されるスクリプト、および関連ファイルのセット。ジョブ実行可能ファイルは、エンジン層のプロジェクト・ディレクトリーに保管される。「*ジョブ (job)*」および「*ジョブデザイン (job design)*」も参照。

ジョブ・パラメーター (job parameter) ジョブデザインのさまざまな時点で使用でき、ジョブの実行中にジョブの処理に対して動的に影響を与えるために指定変更できる処理変数。ジョブ・パラメーターは、ほとんどの場合、パス、ファイル名、データベース接続情報、またはジョブ・ロジックに対して使用される。「*ジョブ (job)*」および「*ジョブデザイン (job design)*」も参照。

ジョブ実行 (job run) ジョブの特定の実行。ジョブを複数回実行して、複数のジョブ実行を生成することができる。

ジョブ・シーケンス (job sequence) 「*シーケンス・ジョブ (sequence job)*」を参照。

層 (layer) 「*層 (tier)*」を参照。

リンク (link) ジョブで各ステージを結合するデータ・フローの表現。リンクは、データ・ソースを処理ステージに接続して、処理ステージを相互に接続し、処理ステージをターゲット・システムにも接続する。リンクのタイプには、入力リンク、出力リンク、参照リンク、および拒否リンクがある。

ルックアップ・テーブル (lookup table) (1) ジョブが参照情報を取得するために使用するキー値を持つデータ・ソース。
(2) 1 つ以上の入力値を 1 つ以上の出力値にマップするために使用されるデータベース表。

マスター・スキーマ定義 (master schema definition) 選択されたデータから生成される推論プロパティーの物理モデル。メタデータの元の定義ではなく、データの推論を反映している。

Match Designer データベース (Match Designer database) InfoSphere QualityStage によって生成されたマッチ・テスト合格の結果を格納するデータベース。

メタデータ (metadata) データの特性を記述するデータ。

メタデータ・サービス (metadata services)F 共通機能 (インポートとエクスポートなど) を InfoSphere Information Server スイートの他の製品モジュールに提供して、すべての層にインストールされる共有コンポーネントのセット。

メトリック (metric) データ・ルール、ルール・セット、およびその他のメトリックに適用できる計測を開発するために定義する式。

自然キー分析 (natural key analysis) キーと関係の機能の一部となっているタイプの分析。この分析は、自然キー候補を検出するためにデータ表を評価する。

ノード (node) 構成ファイル内で仮想名と物理リソース (サーバー、ディスク、プールなど) に関連する詳細情報によって定義される論理処理単位。

Null 可能性 (nullability) Null 値が許可されるかどうかを決定する属性。

オペレーショナル・メタデータ (operational metadata) 発生するイベントとプロセス、およびジョブ実行時に影響を受けるオブジェクトを記述するメタデータ。

オペレーショナル・データベース (operations database) Information Server リポジトリーの 1 コンポーネントであり、オペレーショナル・メタデータと、InfoSphere Information Server スイートの製品モジュールでジョブが実行されるときに使用されたシステム・リソースに関する情報の両方を保管する。「*オペレーショナル・メタデータ (operational metadata)*」および「*Information Server リポジトリー (Information Server repository)*」も参照。

演算子 (operator) パラレルエンジンの一部を成すランタイム・オブジェクト・ライブラリーであり、対応するステージで定義されているようにロジックを実行する。「*ステージ (stage)*」、「*コネクター (connector)*」、および「*ジョブ実行可能ファイル (job executable)*」も参照。

出力リンク (output link) ステージに接続され、一般に処理済みデータをステージから移動するリンク。「*リンク (link)*」も参照。

パック (pack) IBM InfoSphere DataStage の機能を拡張するコンポーネントの集合。

パラレルジョブ (parallel job) パラレルエンジンでコンパイルされて実行され、パラレル処理システム機能 (データのパイプライン化、パーティション化、および分散実行など) をサポートするジョブ。「*ジョブ (Job)*」も参照。

パラレルエンジン (parallel engine) パラレルジョブを実行する、InfoSphere Information Server エンジンのコンポーネント。
以前は「*PX エンジン (PX engine)*」と呼ばれていた。

プラグイン (plug-in) データ・ソースへの接続に使用されるタイプのステージ。ただし、パラレル処理機能はサポートしない。「*コネクター (connector)*」も参照。

1 次キー分析 (primary key analysis) キーおよび関係の機能の一部となっているタイプの分析。この分析は、1 次キー候補を検出するためにデータ表を評価する。

処理ノード (processing node) システム内のジョブが実行される論理ノード。構成ファイルで、システム内の物理ノードごとに 1 つの処理ノードを定義するか、物理ノードごとに複数のノードを定義することができる。

プロジェクト (project) データ統合、データ・プロファイリング、品質モニターなどのために提供、作成、または維持されるオブジェクトを編成してセキュリティーを提供するコンテナー。

PX エンジン (PX engine) 「*パラレルエンジン (parallel engine)*」を参照。

参照リンク (reference link) ルックアップ・テーブルが存在する場所を定義する、Transformer または Lookup ステージの入力リンク。
「*リンク (link)*」も参照。

参照表 (reference table) データ分析のために作成または使用するデータ表。

参照整合性 (referential integrity) 外部キー分析の後で実行され、外部キー候補が関連する 1 次キーの値と一致することを確認するタイプの分析。

拒否リンク (reject link) ステージがレコードを処理しているときにエラーを識別して、拒否されたレコードをターゲット・ステージに経路指定する出力リンク。
「*リンク (link)*」および「*出力リンク (output link)*」も参照。

関係 (relationship) 1 つの表の行、または 2 つの表の行の間の定義済みの接続。関係は、参照制約の内部表現である。

リポジトリー (repository) データおよびその他のアプリケーション・リソースのための永続的なストレージ領域。

レビュー (review) データ分析結果を評価するプロセス。レビュー時に、分析中に行われた推論に同意するか、拒否する。

ルール・セット定義 (rule set definition) データ・ルール定義の集合。

シーケンス・ジョブ (sequence job) 指定された順序で他のジョブを実行するジョブ。

サーバー・エンジン (server engine) サーバー・ジョブおよびジョブ・シーケンスを実行する、InfoSphere Information Server エンジンのコンポーネント。
以前は「*DS エンジン (DS engine)*」と呼ばれていた。

サーバー・ジョブ (server job) サーバー・エンジンでコンパイルされて実行されるジョブ。

サービス層 (services tier) InfoSphere Information Server スイートおよび製品モジュールのアプリケーション・サーバー、共通サービス、および製品サービスと、それらのコンポーネントがインストールされている 1 台以上のコンピューター。

ステージ (stage) ジョブデザインの 1 エレメントであり、データ・ソース、データ処理ステップ、またはターゲット・システムを記述して、データを入力リンクから出力リンクに移動する処理ロジックを定義する。ステージは、ステージ・タイプの構成済みインスタンスである。「*ステージ・タイプ (stage type)*」および「*ジョブデザイン (job design)*」も参照。

ステージ・タイプ (stage type) ステージの機能、ステージのパラメーター、およびステージが実行時に使用するライブラリーを定義するオブジェクト。「*ステージ (stage)*」も参照。

システム推論 (system inference) 「*推論 (inferences)*」を参照。

表分析 (table analysis) 1 次キー分析と、複数列の 1 次キーと潜在的な重複値の評価で構成される分析プロセス。

層 (tier) コンポーネントと、これらのコンポーネントがインストールされているコンピューターの論理グループ。
以前は「*層 (layer)*」と呼ばれていた。

トリガー (trigger) シーケンス・ジョブのアクティビティーを結合するワークフロー・タスク間の依存関係の表現。
アクティビティーには、通常は 1 つの入力トリガーと複数の出力トリガーがある。「*シーケンス・ジョブ (sequence job)*」も参照。

固有性しきい値 (uniqueness threshold) 列が固有であるかどうかを推測する列分析の設定。

ユーザー分類コード (user classification code) データ・フィールドに割り当てられたユーザー定義のコード。例えば、コード FIN は財務データを参照する場合がある。

妥当性評価 (validity assessment) 有効値と無効値についてデータ列を評価するデータ分析プロセス。

仮想列 (virtual column) 既存の列であるかのように分析できる複数の列の連結。

省略語および頭字語

ACR	自動クライアント転送
API	アプリケーション・プログラミング・インターフェース
BI	ビジネス・インテリジェンス
CLI	コマンド・ライン・インターフェース
CSV	コンマ区切り値
DEV	開発
DNS	ドメイン・ネーム・システム
DR	災害復旧
DS	DataStage
DSN	データ・ソース名
ETL	抽出、変換、およびロード
GFS	Global File System
GPFS	General Parallel File System
GUI	グラフィカル・ユーザー・インターフェース
HADR	高可用性災害復旧
IADB	Information Analyzer データベース
IDE	統合開発環境
IP	インターネット・プロトコル
ISD	InfoSphere Services Director
ITSO	International Technical Support Organization
J2EE	Java 2 Enterprise Edition
JVM	Java 仮想マシン
LDAP	Lightweight Directory Access Protocol
LSF	Load Sharing Facility
MPP	超並列処理
MS	Microsoft (省略語は本書の図で使用されています)
MWB	Metadata Workbench
NAS	Network Attached Storage
ODBC	Open Database Connectivity
PRE-PROD	疑似本番
PROD	本番
PVU	プロセッサー・バリュー・ユニット
OGE	Oracle Grid Engine
QS	QualityStage
QSDB	QualityStage データベース
RAC	Real Application Clusters
RAM	ランダム・アクセス・メモリー
RDBMS	リレーショナル・データベース管理システム
REST	Representational State Transfer
SAN	ストレージ・エリア・ネットワーク
SGE	Sun Grid Engine
SMP	対称型マルチプロセッサー
SQL	構造化照会言語
SSH	セキュア・シェル
URL	Uniform Resource Locator
WLM	Workload Manager
XML	Extensible Markup Language

関連資料

このセクションにリストされている資料は、本資料で扱うトピックをさらに詳しく検討するのに特に適していると考えられます。

IBM Redbook

下記の IBM Redbook 資料には、本書のトピックに関する追加情報が記載されています。このリストに掲載されている資料の中にはソフトコピーでのみ利用できるものがあることに注意してください。

- 「*Information Server: Installation and Configuration Guide*」(REDP-4596)
- 「*Metadata Management with IBM InfoSphere Information Server*」(SG24-7939)

上記の資料およびその他の Redbook、Redpaper、Web 文書、ドラフトや資料の検索、表示、ダウンロード、または注文は、次の Web サイトから可能です。

ibm.com/redbooks

オンライン・リソース

次の Web サイトも、IBM ソフトウェアおよびハードウェア製品の用語および定義に関連しています。

http://www.ibm.com/software/globalization/terminology/

IBM の支援

IBM サポートおよびダウンロード

ibm.com/support

IBM Global Services

ibm.com/services

Information Server デプロイメント・アーキテクチャー

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Information Server デプロイメント・アーキテクチャー

小規模から始めて大規模に拡張する道のり

メタデータに関する考慮事項について

グリッドおよび SMP 環境でのデプロイメント

典型的なデプロイメント・アーキテクチャーでは、複数の製品、環境、およびサーバーにまたがる共有メタデータ・プラットフォームをフルに活用する上で課題が生じます。データ・プライバシーと機密保護の要件により、さらに複雑さの度合いが高くなります。IBM InfoSphere Information Server は、異機種混合システムで信頼できる情報を配信するための包括的なメタデータ主導型プラットフォームを提供します。

この IBM Redbook 資料では、プラットフォームの共有メタデータ機能を使用して、開発ライフサイクル、データ・プライバシー、機密保護、高可用性、およびパフォーマンスの要件をサポートする代表的な論理インフラストラクチャー・トポロジーにおいて Information Server コンポーネントのデプロイメントを成功させるためのガイドラインと基準を提示しています。本書は、提示するガイドラインに基づいて、特定のユース・ケースに対応した適切なデプロイメント・アーキテクチャーを決定するために情報要件を評価する上で役立ちます。また、ビジネス目標を達成するために Information Server 製品モジュールおよびコンポーネントの機能を効果的に使用する上でも役立ちます。

本書は、IBM InfoSphere Information Server の情報統合機能の完全スイートを提供する責任がある IT アーキテクト、情報管理と情報統合のスペシャリスト、およびシステム管理者を対象としています。

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

実際的経験に基づいて技術情報を構築

IBM Redbook は、IBM International Technical Support Organization によって作成されています。世界中の IBM、お客様、およびパートナー様の専門家が現実的なシナリオに基づいてタイムリーな技術情報を形成しています。お客様の環境で IT ソリューションをさらに効果的に実装する上で役立つ具体的なアドバイスが掲載されています。

詳細については、次の Web サイトをご覧ください。
ibm.com/redbooks

ISBN 073843728X

SG88-4072-00
（英文原典 :SG24-8028-00）