IBM Security Directory Integrator
バージョン 7.2.0.1

スタートアップ・ガイド

GI88-4235-03
(英文原典：GI11-9325-03)

IBM Security Directory Integrator
バージョン 7.2.0.1

スタートアップ・ガイド

GI88-4235-03
(英文原典：GI11-9325-03)

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、 115 ページの『特記事項』に記載されている情報をお読みください。

注: 本書は、*IBM Security Directory Integrator* バージョン 7.2.0.1 ライセンス・プログラム (5724-K74)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： GI11-9325-03
IBM Security Directory Integrator
Version 7.2.0.1
Getting Started Guide

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

図

本書について

IBM® Security Directory Integrator スタートアップ・ガイドを使用して、Security Directory Integrator の基本概念と、効率的なデータ統合ソリューションの設計の背景を把握します。

本書では、IBM Security Directory Integrator に関する概念的な情報を説明し、製品の使用を開始する際に役立つ例を提供しています。

資料および用語集へのアクセス

以下は英語のみの対応となります。オンラインでアクセス可能な IBM Security Directory Integrator バージョン 7.2.0.1 ライブラリーおよび関連資料の説明をお読みください。

このセクションには、以下が含まれています。

- 『IBM Security Directory Integratorライブラリー』の資料のリスト。
- viii ページの『オンライン資料』へのリンク。
- ix ページの『IBM Terminology Web サイト』へのリンク

IBM Security Directory Integratorライブラリー

IBM Security Directory Integrator ライブラリーでは、以下の資料を入手できます。

- *IBM Security Directory Integrator バージョン 7.2.0.1 Federated Directory Server 管理ガイド*

  Federated Directory Server コンソールを使用したデータ統合ソリューションの設計、実装、および管理について説明しています。System for Cross-Domain Identity Management (SCIM) プロトコルおよびインターフェースを使用した ID 管理についても説明しています。

- *IBM Security Directory Integrator バージョン 7.2.0.1 スタートアップ・ガイド*

  IBM Security Directory Integrator の解説および概要です。対話の作成の例と、IBM Security Directory Integrator の実践学習を含んでいます。

- *IBM Security Directory Integrator バージョン 7.2.0.1 ユーザーズ・ガイド*

  IBM Security Directory Integrator の使用方法が記載されています。Security Directory Integrator デザイナー・ツール (構成エディター) を使用したソリューションの設計や、コマンド行からの既製ソリューションの実行について説明しています。また、インターフェース、概念、および AssemblyLine の作成に関する情報も記載されています。

- *IBM Security Directory Integrator バージョン 7.2.0.1 インストールおよび管理者ガイド*

インストール、旧バージョンからのマイグレーション、ロギング機能の構成、および IBM Security Directory Integrator のリモート・サーバー API の基礎となるセキュリティー・モデルについて記載されています。ソリューションのデプロイおよび管理方法が含まれています。

- *IBM Security Directory Integrator バージョン 7.2.0.1 リファレンス・ガイド*

  IBM Security Directory Integrator の個々のコンポーネント (コネクター、関数コンポーネント、パーサー、オブジェクトなど) に関する詳細情報が記載されています。

- *IBM Security Directory Integrator バージョン 7.2.0.1 Problem Determination Guide*

  問題の識別および解決を支援する IBM Security Directory Integrator のツール、リソース、および技法に関する情報が記載されています。

- *IBM Security Directory Integrator バージョン 7.2.0.1 Message Guide*

  IBM Security Directory Integrator に関連付けられたすべての情報、警告、およびエラー・メッセージのリストが記載されています。

- *IBM Security Directory Integrator バージョン 7.2.0.1 パスワード同期プラグイン*

  5 つの IBM Password Synchronization Plug-ins (Windows 用 Password Synchronizer、Sun Directory Server 用 Password Synchronizer、IBM Security Directory Server 用 Password Synchronizer、IBM Domino 用 Password Synchronizer、UNIX および Linux 用 Password Synchronizer) それぞれのインストールおよび構成について詳細に説明されています。また、LDAP パスワード・ストアと JMS パスワード・ストアの構成手順についても説明します。

- *IBM Security Directory Integrator バージョン 7.2.0.1 リリース・ノート*

  IBM Security Directory Integrator に関する新規機能および最新情報で、本書に記載していないものを掲載しています。

オンライン資料

IBM では、製品のリリース時および資料の更新時に、以下の場所に製品資料を掲載しています。

IBM Security Directory Integrator ライブラリー
製品資料サイト (http://www-01.ibm.com/support/knowledgecenter/SSCQGF/welcome) には、ライブラリーのウェルカム・ページとナビゲーションが表示されます。

IBM Security Systems Documentation Central
IBM Security Systems Documentation Central には、すべての IBM Security Systems 製品ライブラリーのアルファベット順のリストと、各製品の特定バージョンのオンライン資料へのリンクが掲載されています。

IBM Publications Center
IBM Publications Center サイト (http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) には、必要なすべての IBM 資料を見つけるのに役立つカスタマイズ検索機能が用意されています。

関連情報

IBM Security Directory Integrator の関連情報は以下の場所で入手できます。

- IBM Security Directory Integrator では、Oracle の JNDI クライアントを使用しています。JNDI クライアントについては、「*Java Naming and Directory Interface™ Specification*」(http://download.oracle.com/javase/7/docs/technotes/guides/jndi/index.html) を参照してください。
- IBM Security Directory Integrator に関する疑問点を解決するために有用な情報が https://www-947.ibm.com/support/entry/myportal/over-accesspubsview/software/security_systems/tivoli_directory_integrator に記載されています。

IBM Terminology Web サイト

IBM Terminology Web サイトは、製品ライブラリーの用語を 1 つのロケーションに統合したものです。Terminology Web サイトには、http://www.ibm.com/software/globalization/terminology からアクセスできます。

アクセシビリティー

アクセシビリティー機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。この製品では、インターフェースを音声化およびナビゲートするための支援テクノロジーをご利用になれます。また、マウスの代わりにキーボードを使用して、グラフィカル・ユーザー・インターフェースのすべての機能を操作できます。

詳しくは、「*Directory Integrator の構成*」のアクセシビリティーに関する付録を参照してください。

技術研修

以下は英語のみの対応となります。技術研修の情報については、IBM Education Web サイト (http://www.ibm.com/software/tivoli/education) を参照してください。

サポート情報

以下は英語のみの対応となります。IBM サポートは、コード関連の問題、およびインストールまたは使用方法に関する短時間の定型質問に対する支援を提供します。IBM ソフトウェア・サポート・サイトには、http://www.ibm.com/software/support/probsub.html から直接アクセスできます。

「トラブルシューティング」では、以下が詳細に説明されています。

- IBM サポートに連絡する前に収集する情報。
- IBM サポートに連絡するためのさまざまな方法。
- IBM Support Assistant の使用方法。
- 自分で問題を切り分け、修正するための説明および問題判別の資料。

適切なセキュリティーの実践に関する注意事項

IT システムのセキュリティーでは、企業内および企業外からの不適切なアクセスの防止、検出、およびそれらのアクセスへの対応により、システムおよび情報を保護する必要があります。不適切なアクセスにより、情報が改ざん、破壊、盗用、または悪用されたり、ご使用のシステムの損傷または他のシステムへの攻撃のための利用を含む悪用につながる可能性があります。完全に安全と見なすことができる IT システムまたは IT 製品は存在せず、また単一の製品、サービス、またはセキュリティー対策が、不適切な使用またはアクセスを防止する上で、完全に有効となることもありません。IBM のシステム、製品およびサービスは、包括的なセキュリティーの取り組みの一部となるように設計されており、これらには必ず追加の運用手順が伴います。また、最高の効果を得るために、他のシステム、製品、またはサービスを必要とする場合があります。IBM は、システム、製品、またはサービスが、悪意のある行為または不正な行為から影響を受けないこと、またはこれらの行為がお客様の企業に影響を与えないことを保証しません。

第 1 章 概要

本書は、単純なシステムについて簡単に紹介したものです。ただし、ここでは*単純*という語を、最も肯定的でパワフルな意味合いで使っていることに注意してください。すなわち、複雑な問題を理解するための最善の策は、問題をより単純で扱いやすい個々の部分に細分化し、それらの構成部分を 1 つ 1 つマスターすることだからです。 Divide-and-Conquer 手法、分割統治法などと言います。これは、日常的な問題を解決するために無意識に使用されている技法ですが、オフィス全内、企業内、あるいは地球全体にわたる技術的情報交換においても同じく有効です。

情報をすぐにファイル、ディレクトリー、データベース、または IBM Notes から取り出し、このデータを他の場所に転送する場合は、付録 93 ページの『EasyETL ガイド』へ直接スキップしてください。この機能を使用すると、最初に中核となる概念を学ばなくてもIBM Security Directory Integrator の能力を利用することができます。 ソースとターゲットを選択してから「実行」を押し、データ・フローを確認します。これに対して、データの読み取り、フィルタリング、品質向上、変換および移動を行う方法について、より詳細に制御したい場合は、以降の説明を読み進めてください。 93 ページの『EasyETL ガイド』は後の楽しみにとっておきましょう。

IBM Security Directory Integrator[1] は、最も複雑な統合問題でさえも、次の 3 つの基本部分に分解できるという前提に基づいて設計され、構築されています。

- 通信に関係するシステム (データ・ソース とも呼ばれる)
- それらのシステム間のデータ・フロー
- データ・フローを起動するイベント

IBM Security Directory Integrator を使用すると、統合の問題についてのこのアトミックな理解からソリューションを直接的に導き出すことができます。ソリューションは、一度に 1 フローずつ、フィードバックと検証を繰り返しながら、漸進的に構築できます。このアプローチによって、統合プロジェクトの見積もりや計画が容易になります。場合によっては、インプリメントする個々のデータ・フローのコスト計算をして決定するだけで済むこともあります。 また、1 つのタスクを実行可能なステップごとに着実に進めていくことで、利害関係者に対して定期的に進捗状況を示すことができます。

IBM Security Directory Integrator は、データ・ソース間の技術的な相違を抽象化して吸収することにより、開発作業をさらに促進します。そのため、業務要件に集中してより多くの時間を費やすことができるようになります。

Eclipse の機能を活用することによって、IBM Security Directory Integrator 開発環境は包括的で拡張性の高いものになっています。統合プロジェクトは、結果的にコンポーネントおよびビジネス・ロジックのライブラリーとなるため、それらを再利用して新しい課題に対処することができます。その結果、組織内に分散するすべての

1. 名前に惑わされないでください。IBM Security Directory Integrator はディレクトリー関連の処理だけでなく、すべての主要なデータ・ストア、トランスポート、プロトコル、および API をサポートします。その中にはもちろん LDAP ディレクトリーも含まれます。

開発チームが IBM Security Directory Integrator の資産を共用できるため、一貫して統合および管理されたインフラストラクチャーに即時に適合する独立プロジェクト (さらにはポイント・ソリューションさえも) が生まれます。

本書では、上で述べた、単純化して解決する方法を紹介します。ここからはまた、洗練された単純な IBM Security Directory Integrator のツール・セットを活用する第一歩を踏み出すことになります。具体的には、次の 2 つのプログラムです。

- *構成エディター* (または略して「CE」) と呼ばれる開発環境
- サーバー と単純に呼ばれるランタイム・エンジン

CE を使用して IBM Security Directory Integrator ソリューションを作成し、1 つ以上のサーバーを使用して稼働されます。これらのプログラムは協調して動作し、シームレスなユーザー・エクスペリエンスを実現すると同時に、異なるプラットフォームでの作業を可能にします。例えば、手元のラップトップでソリューションを開発しながら、それをメインフレームでリモート実行してテストおよびデバッグを行うことができます。

JavaScript によるスクリプト記述

上で説明したように、IBM Security Directory Integrator は統合ソリューションの素早い作成を可能にします。 ただし、組み込みの自動化機能を、独自のカスタム処理やフローの動作によって拡張するためには、スクリプトのスニペットを記述する必要があります。

スクリプト記述は、JavaScript で行われます。また、IBM Security Directory Integrator には、高速で信頼性の高いスクリプト環境を提供する IBM JSEngine が含まれています。そのため、JavaScript 言語の中核となる部分を理解して使用する必要があります。JavaScript の学習用としては、いくつかの優れたオンラインおよびハードコピーのリソースがあります。IBM Security Directory Integrator ニュースグループおよび Web サイトを調べて、推奨リソースと関連リンクを参照してください。

IBM Security Directory Integrator におけるスクリプト記述の詳細については、「*Directory Integrator の構成*」を参照してください。

IBM Security Directory Integrator のインストール

IBM Security Directory Integrator のインストールは数分間で完了し、ソリューションの作成、テスト、配置を即座に開始できます。システムは、Microsoft Windows、IBM AIX®、IBM System z®、ならびにいくつかの UNIX および Linux 環境を含む広範なプラットフォーム上で動作します。

IBM Security Directory Integrator のインストールには、以下の 3 つのパスが関係します。インストーラーは、最初の 2 つのパスを指定するように求めてきます。

1. *インストール・ディレクトリー* - ここには、プログラム・ファイルおよび各種のツールを起動するために使用するバッチ・ファイルまたはスクリプトが保存されています。
2. *ソリューション・ディレクトリー* - これは、「SolDir」と略記されることが多く、IBM Security Directory Integrator の実行時には現行フォルダーとなります。Config Editor 開発環境 (`ibmditk`) およびサーバー (`ibmdisrv`) のスタートアップ用バッチ・ファイルおよびスクリプトが、いずれもソリューション・ディレクト

リーへのディレクトリー変更コマンドで始まっていることに注意してください。その結果、ソリューションで使用されているすべての相対パスは、ソリューション・ディレクトリーを基点として展開されます。

3. ワークスペース・フォルダー - ここには、プロジェクト・ファイルおよびリソース・ファイル[2]が保存されています。デフォルトでは、このフォルダーが「workspace」という名前でソリューション・ディレクトリー内に作成されます。

IBM Security Directory Integrator のインストールについて詳しくは、「*インストールと管理*」の『*IBM Security Directory Integrator のインストールに関する説明*』を参照してください。

チュートリアル・ファイルのインストール

本書のチュートリアルの練習を実行するためには、サポート用のデータ・ファイルが必要です。それらのファイルは、IBM Security Directory Integrator インストール・ディレクトリーのサブフォルダー examples/Tutorial の中にあります。例えば、標準的な Windows インストール環境では、これらのファイルが次のディレクトリーに格納されます。

```
C:¥Program Files¥IBM¥TDI¥V7.2¥examples¥Tutorial
```

「チュートリアル」ディレクトリーには、以下の各ファイルが格納されています。

- CreatePhoneDB.assemblyline
- index.html
- OtherPage.html
- People.csv
- PhoneNumbers.xml
- readme.txt
- Return web page.script

注: 前のセクションで説明したとおり、インストーラーはソリューション・ディレクトリーの場所を指定するように求めてきます。これはプロジェクト・ファイルとリソース・ファイルが格納される場所であり、通常、ホーム・エリアの下にあるMy Documents¥TDI というサブディレクトリーです。

構成エディターのツールからアクセスしやすくするために、チュートリアル・フォルダーをソリューション・ディレクトリーにコピーします。

単純化による解決

本セクションでは、データ統合ソリューションの設計にあたってどこから着手したらよいかを説明します。ここで説明する設計手法は段階的に進む方法ですが、大規模なものも含め、あらゆる規模の統合やシステム展開プロジェクトに適用することができます。

2. これらの用語については、 11 ページの『第 2 章 IBM Security Directory Integratorの紹介』で説明します。

IBM Security Directory Integrator を利用したデータ統合ソリューションは、次に示す手法で段階的に設計します。

- 問題をより小さくて扱いやすいように分割し、複雑さを減らす。
- ソリューション全体の中で、できれば 1 週間から 2 週間で完了できる部分から着手する。
- ソリューション全体の中で、自動的に成果を出せる部分から着手する。

複雑なプロジェクトの処理

大規模の統合およびシステム・デプロイメントのプロジェクトを要約して処理する最良のアプローチは、問題をより小さくて扱いやすいように分割して、複雑さを減らすことです。これが完了したら、ソリューション全体の一部分、できれば独立して展開できる部分から着手してください。これにより、残りの部分に取り組んでいる間に、その部分が早くも投資への見返りを生み出してくれます。

処理しようとしている部分を他と分離したら、通信の基本単位 (データ・フローそのもの) に焦点を当て、さらに単純化します。これで、インプリメントを開始する準備が整います。統合開発は、IBM Security Directory Integrator 構成エディター (「CE」と略記) を利用して「試行/テスト/改良」という一連のサイクルを通じて実施されるため、プロセスが反復的で探究的なものになります。これは、お客様自身のインストール・システムについてさらに多くのことを発見するのに役立つだけでなく、問題への理解とその問題がインフラストラクチャーに与える影響への理解が深まるにつれて、統合ソリューションを発展させることを可能にします。

関連トピック

以下のトピックでは、IBM Security Directory Integrator で AssemblyLine を使用したデータ変換が可能になる仕組みを説明します。

- 『カーネル/コンポーネントのアーキテクチャー』
- 5 ページの『項目属性値データ・モデル』
- 7 ページの『データ・フロー = AssemblyLine』

カーネル/コンポーネントのアーキテクチャー

IBM Security Directory Integrator の基本的な性質は、カーネル/コンポーネントの設計方式に由来します。

ここで言うカーネル とは、統合ソリューションの素早い作成を可能にし、それらを駆動するための自動実行ロジックを提供するラピッド・インテグレーション開発 (RID) フレームワークのことです。通常はハンド・コーディングを必要とする (そのため無視されることが多い) ログ/トレース・モジュール、接続復旧、変更検出、エラー処理、および外部管理 API のような機能が、最も単純なデータ・フローにおいても即座に利用可能になります。

このカーネルの汎用機能に加え、IBM Security Directory Integrator はデータ・ソース固有のコンポーネントのセットを提供します。ヘルパー・オブジェクトと呼ばれるこれらのコンポーネントは、さまざまなデータ・ソースとの対話における技術的な詳細を抽象化して吸収します。最も使用頻度の高いコンポーネントは、コネクター とパーサー の 2 つです。

コネクターは、広範なデータ・ソースに対して接続性を提供すると同時に、基盤になる組織構造に関係なく、構造化データに特有の処理を行います。コネクターには、(例えば IP ポートにバインドして着信接続を待機する、あるいはディレクトリー、データベース、またはファイルに生じる変更を「listen」するなどして) イベント・ハンドラーの役割を果たすものもあります。

一方パーサーは、非構造化データ (つまり、バイト・ストリーム) の処理に使用されます。例えば、ファイル、POP3/SMTP 電子メール、MQ メッセージ、IP ポート間のデータ・ストリームなどに含まれるデータです。

IBM Security Directory Integrator は、それぞれ特定のシステム、サービス、API、トランスポート、またはフォーマットを処理するために設計されたコネクターおよびパーサーの拡張可能なライブラリーを提供します。IBM Security Directory Integrator のコンポーネントは交換可能なため、テスト・データ (例えばテキスト・ファイル) に基づいてソリューションをビルドした後、使用するコネクターを簡単に交換して、ソリューションを実際のソースに接続し、検証および配置を行うことができます。

また、IBM Security Directory Integrator のコンポーネントは簡単に使用することができ、ビルドおよび拡張も容易です。コミュニティー Web サイトから新規コンポーネントをダウンロードしたり、Java™ で独自のコンポーネントを記述したり、CE 環境で直接スクリプトを使用して対話的にコンポーネントのビルドおよびテストを行うことによって、カスタム・データ・ソースおよびカスタム・サービスの処理のためにライブラリーを拡大することができます。

項目属性値データ・モデル

データを編成して格納する方法は、システムによって大幅に異なっています。

- データベースでは通常、固定列数と共に、情報が行単位で格納されます。各列は、レコードの 1 つの値を保持しています。
- ディレクトリーは、可変個数の属性を含めることのできるオブジェクト指向の項目を保持します。同様に 0 か、1 つもしくは複数の値を持ちます。[3]
- IBM Domino データベースは、各々が単一の値か複数の値を持つものとして定義できるフィールドを構成する文書を収納します。
- なお、データ内容をノードやオブジェクト、レコード、フォーマット済みバイト・ストリーム、あるいはキー値セットとして表現しているシステムもあります。

すべての関係者にとって重要であるコミュニケーションのためにはデータ形式は互換性を持っている必要があり、または関連する各システムに適するように変換される必要があります。これはデータ・マーシャリング と呼ばれ、統合処理のスペシャリストがしばしば最初に直面する困難な問題であって、克服するにはかなり大きなプロジェクト・リソースがすぐに費やされてしまいます。 IBM Security Directory Integrator コネクターは、ソースに特化したタイプを一貫して正規表現に自動変換することで、これを実施します。個々のデータ値は、同様の方法で表現されている類似の固有のタイプごとに関連する Java オブジェクトに変換されます。例えば、ファ

3. 複数の電子メール・アドレスを所有していて、そのすべての電子メール・アドレスは企業の従業員ディレクトリー中の「メール」とタイトルされた複数の値を持つ属性で収納できる、という事実を考えてみてください。

イルから読み込まれた行や LDAP 文字列属性、Domino テキスト・フィールド、RDBMS CHAR や VARCHAR カラムはすべて、その関連コネクターによって `java.lang.String` に変換されます。

マーシャルされたこれらの値は、次に IBM Security Directory Integrator によって定義された特殊な Java オブジェクトである属性に蓄積されます。上記のように、ソースの中には同じ属性名の下に複数の値を格納できるものがある一方で、列やフィールドにつき単一の値しか持てないものもあります。 IBM Security Directory Integrator 属性は単一値と複数値の両方のインプリメンテーションをサポートしており、例えばデータベースのヌル可能なカラムを表現するときのように、必要な時には値をまったく持たないことも可能です。

データの 1 つの単位 (レコードやメッセージ、文書など) を形成するすべての属性は、*項目* と呼ばれる別 IBM Security Directory Integrator オブジェクトに集められます。 1 つの項目にはいくつもの属性を収納できます。あるいは、まったく空でもかまいません。

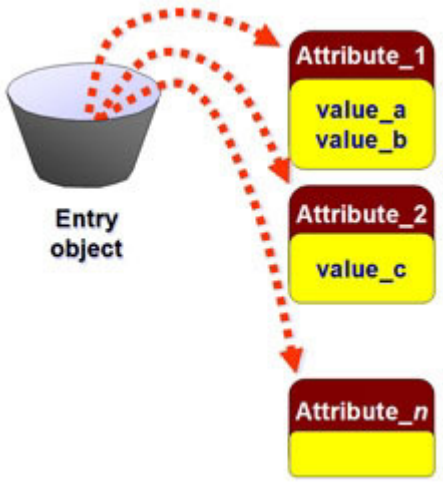

図 1. 項目属性値データ・モデル

各データ・フローは作業項目というプライマリー項目「バケット」を持ちます。コネクターがデータを読み込む度に、属性を作成して作業項目に格納します。出力用に構成されたコネクターはいずれも、変更をターゲット・システムに伝えるために、既に作業項目中で検出されている属性を使用します。

この二段階のアプローチにより、データの転送、変換、フィルタリングおよび品質向上の手法において、ほぼ無限の柔軟性が提供されます。また、出力システムとの接続を考慮する必要がある以前に、まず入力コネクターを使ってデータ・フロー全体を構築し、データが読み込まれて操作される際に CE で対話的に検証することができるということを意味します。

後でわかることですが、IBM Security Directory Integrator 項目はフラットなスキーマを扱う場合とまったく同じくらい容易に、複雑な階層型のデータを扱います。

データ・フロー = AssemblyLine

本書およびその他の文書で、ソリューションにおける個々のデータ・フローは 1 つの IBM Security Directory Integrator *AssemblyLine* として、また、「AL」と短縮されてインプリメントされます。

AL は入力ソースからターゲットまでの一貫した単一のパスを形成するコンポーネント順序リストです。カーネルが提供する組み込み動作はコンポーネントと一体になって、作業項目に持ち込まれたデータを次々と受け渡していきます。

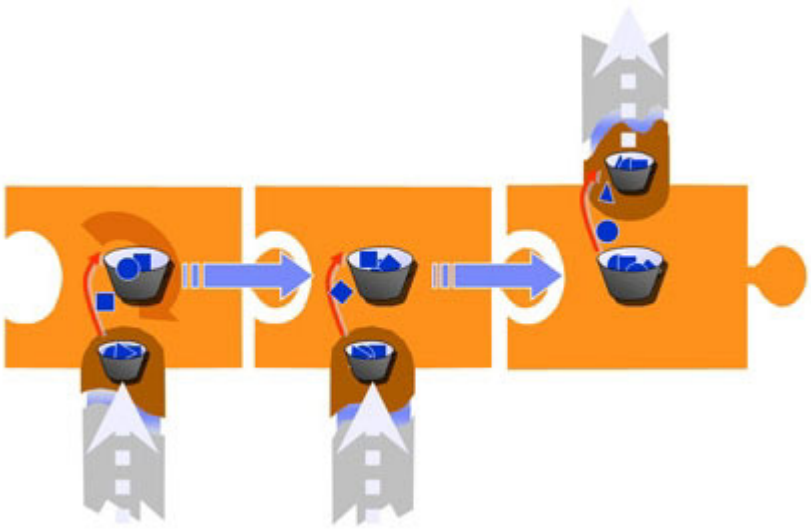

図 2. AssemblyLine の下流へと送られるデータ

「百聞は一見に如かず」と言いますが、上図も例外ではありません。パズルのピース 3 つはコネクターを表し、それらがリンクされて 1 つの AssemblyLine を形成しています。各パズル・ピースの色の濃い「茎」は、コネクターのデータ・ソース特定の部分を強調表示していますが、それは接続されているシステムへのインターフェース で、コネクター・インターフェース (略記「CI」) と言います。各パズル・ピースの残りの色の薄い部分はカーネルの汎用機能であり、どのコンポーネントにも類似した予測可能な方法で処理を行わせ、コンポーネントの相互リンクを可能にし、自動化された動作パターンをカスタマイズするための制御点を提供します[4]。

この図には、さらにいくつかの重要な概念が示されています。例えば、コンポーネントからコンポーネントへとデータを AssemblyLine の下流に送るために使用される上記の作業項目に加えて、この図は各コネクター・インターフェースに付加的に組み込まれる「Java バケット」を示しています。各ローカル項目オブジェクトはその CI による読み取りおよび書き込み操作中にデータをキャッシュするために使用され、*Conn 項目* と呼ばれます。

4. このように、AssemblyLine のすべてのコンポーネントは、IBM Security Directory Integrator のカーネル/コンポーネント・アーキテクチャーを反映します。ユーザー独自のコンポーネントを作成する場合に、インプリメントする必要があるのはコンポーネントのインターフェースだけです。AL「ラッパー」とその豊富な組み込み機能はIBM Security Directory Integrator カーネルのおかげで自動的に有効になります。

ここでは、様々な Conn 項目と AL の作業項目の間のデータの流れを、曲線の矢印で示しています。これらは*属性マップ* で、各矢印は AL への取り込みまたは AL からの出力というデータの移動や変換の一連の規則を示しています。"Conn 項目から作業項目へデータを持ち上げている矢印は、AssemblyLine に取り込むデータを決定しているので、*入力マップ* と呼ばれます。右端にあるパズル・ピースの、Conn 項目から作業項目の方向へと、反対向きに移動しているデータを示す矢印は、*出力マップ*と呼ばれます。

存在する作業項目は常に 1 つのみのため、結果的に AssemblyLine は一度に 1 つの項目、例えば、1 つのデータベース行、1 つのディレクトリー項目、1 つの MQ メッセージなどを処理することになります。これが IBM Security Directory Integrator のもう 1 つの重要な面であり、秒あたり何百、または何千もの項目のサイクルが AssemblyLine で可能ではありますが、[5]ソリューションを設計する際の重要な考慮点になっています。作業を複数の AssemblyLine に拡張することも可能です。これについて、および AL のパフォーマンスを最適化する他の技法については、その他の IBM Security Directory Integrator 文書を参照してください。

始めに

どのような統合プロジェクトでも、始めに当面の問題を図式化してみることが大切です。

紙と鉛筆を使用して、望ましいフローを大まかに描いてみましょう。できあがったスケッチは、タスクの範囲を視覚化するために役立つだけでなく、IBM Security Directory Integrator でそのタスクをインプリメントするための青写真の役割を果たします。

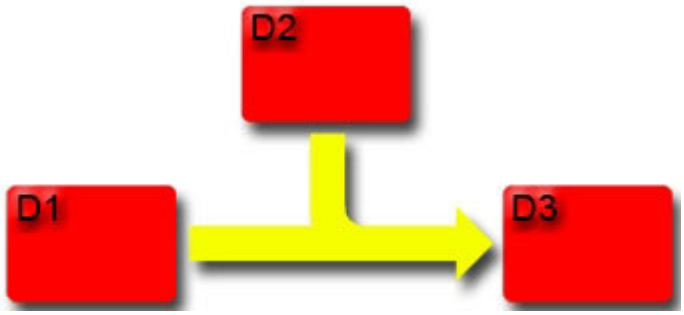

図 3. チュートリアルのシナリオ

IBM Security Directory Integrator ソリューションを作成する際の最初のステップは、データ・ソース間のデータ・フローを、コネクターで構成される AssemblyLine に変換することです。IBM Security Directory Integrator のモットーである「単純化による解決」はできる限り簡単に着手して、段階的にソリューションを確立していくことを表しています。

これを実証するために、最初の AssemblyLine に使用するシナリオ例について考えてみてください。この統合タスクには、D1、D2、および D3 とラベル付けされた 3 つのデータ・ソースが関係しています。このソリューションの目的は、D1 のコンテ

5. パフォーマンスは AssemblyLine の設計と複雑さの度合い、サーバーを稼働させているマシンの構成により決まります。

ンツを D3 に移行して、そのデータに D2 にある値を増加させることです。この要求を AssemblyLine に変換すると、各データ・ソースに 1 つずつ、合計 3 つのコネクターが必要になります。

1. 1 つめのコネクターは、D1 を通して繰り返し フローにデータをフィードします。
2. それに続く 2 つめのコネクターは、D2 の関連するレコードを*ルックアップ*し、D1 から送られてくる値とマージします。
3. 最後に 3 つめのコネクターは、増加したこれらのレコードを D3 に*加算* するように構成されています。

問題全体を一度に攻略するのではなく、IBM Security Directory Integrator はわずか 2 つのコネクターから開始することによって、タスクを単純化することを可能にします。一方のコネクターは D1 の内容を AL に読み取り、もう一方がその値を D3 に書き込みます。 この最小の AssemblyLine が正しく作動するようになれば、D2 へのコネクターで追加の属性を結合し、この AssemblyLine を拡張することができます。これが最初の IBM Security Directory Integrator ソリューションの具体的な作成方法です。本書の残り部分では、このプロセスをステップごとに進めていく方法を説明します。

第 2 章 IBM Security Directory Integratorの紹介

このセクションでは IBM Security Directory Integrator の本質を理解するのに役立つ情報、ならびに開発環境を実践的に体験できるチュートリアルでの練習を提供します。

本製品を理解する最初のステップとして、「構成エディター」または「CE (略称)」と呼ばれる IBM Security Directory Integrator 開発ツールを開始します。

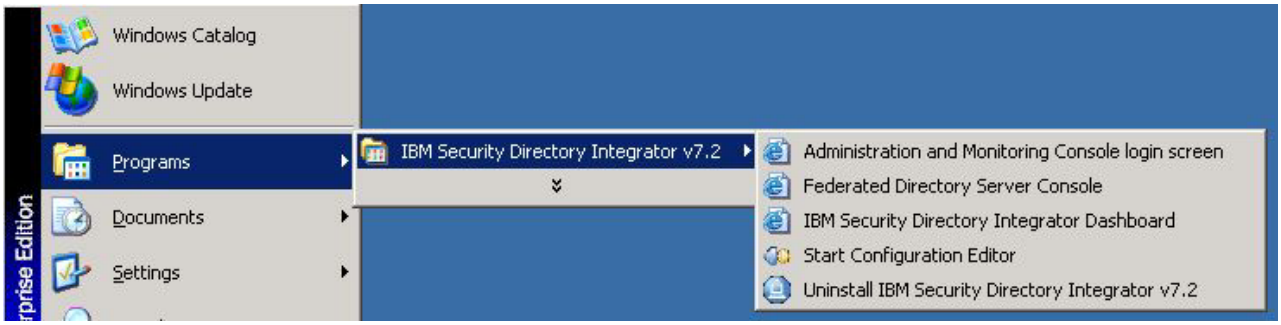

図 4. 構成エディターの開始

初めて CE を開始すると、ワークスペースを指定する次のダイアログが表示されます。

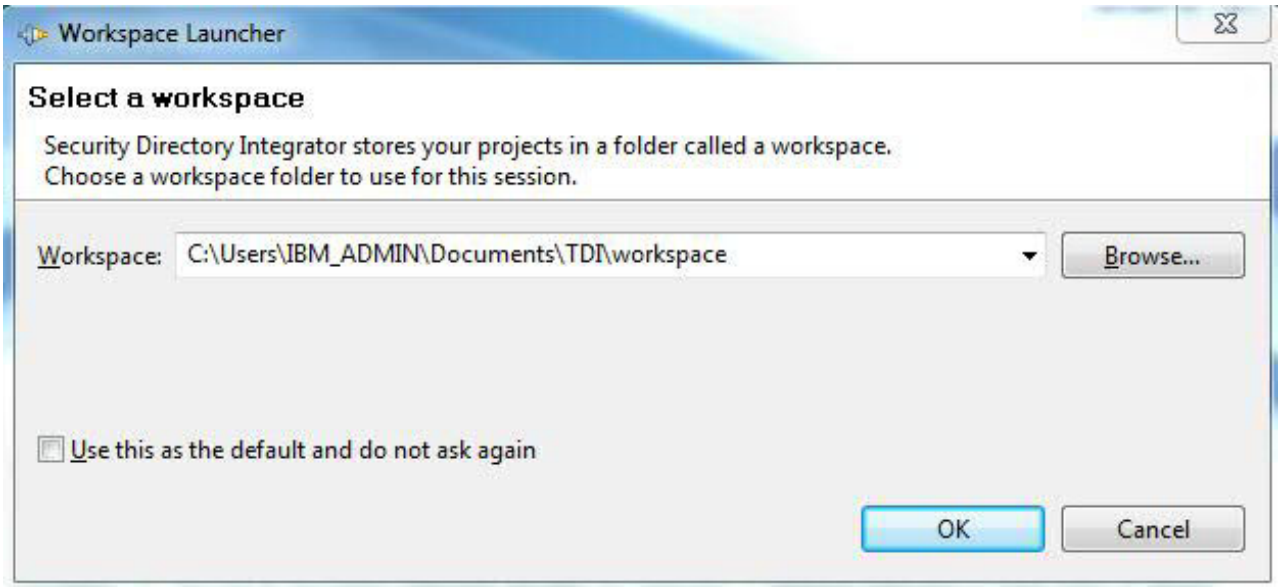

図 5. ワークスペースの選択

ワークスペースは構成エディターがコンポーネントおよび AssemblyLine を含むプロジェクト・ファイルを保管する場所で、通常、ソリューション・ディレクトリーの下にあります。

ワークスペースの位置を確定したら、「**OK**」ボタンをクリックしてください。ウェルカム画面が表示されます。

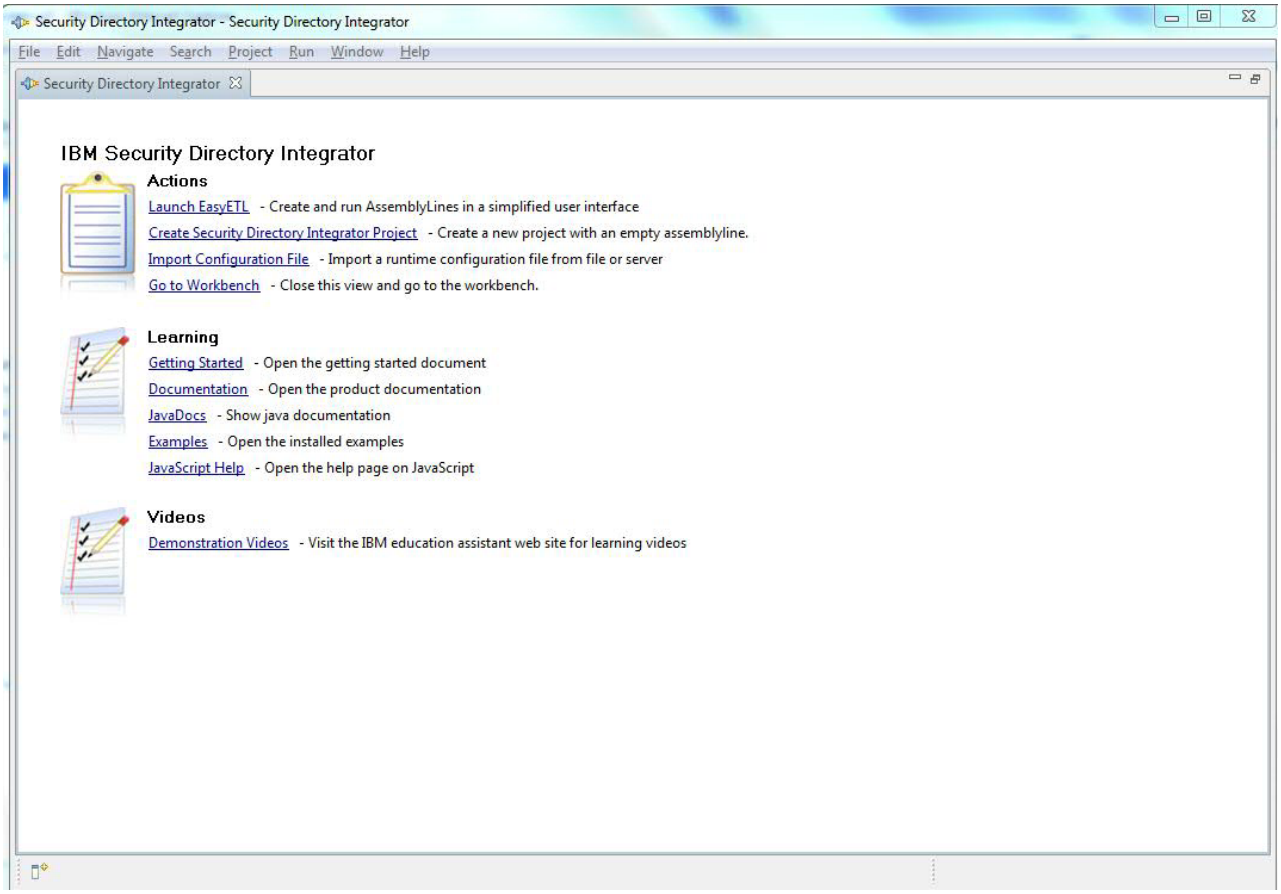

図 6. 構成エディターのウェルカム画面

ウェルカム画面では複数のクイック・スタート・リンクが表示されます[6]。

IBM Security Directory Integrator で統合ソリューションの作成、テスト、および変更を行う場合、プロジェクトの中で作業することになります。プロジェクトは AssemblyLine およびそれを構成するコンポーネントのまとまりで、各プロジェクトはワークスペースのそれぞれのサブフォルダー内に表示されます。プロジェクトを構成する AssemblyLine とコンポーネントは、個々のファイルとして保存され、プロジェクト・フォルダーのサブディレクトリーに、順に配置されます。

このページの上から 2 番目のリンク [7] (*Security Directory Integrator Project*) を作成して、最初のプロジェクトをセットアップします。新規のプロジェクトには名前を付ける必要があります。「チュートリアル」という名前を付けて、「**完了**」をクリックしてください。

6. この画面にはメインメニューの「**ヘルプ**」 > 「**ようこそ**」を選択することでいつでも戻ることができます。

7. 一番上のリンク「EasyETL を起動します」を選択すると、簡素化されたワークベンチが表示されます。詳細は、付録 93 ページの『EasyETL ガイド』を参照してください。

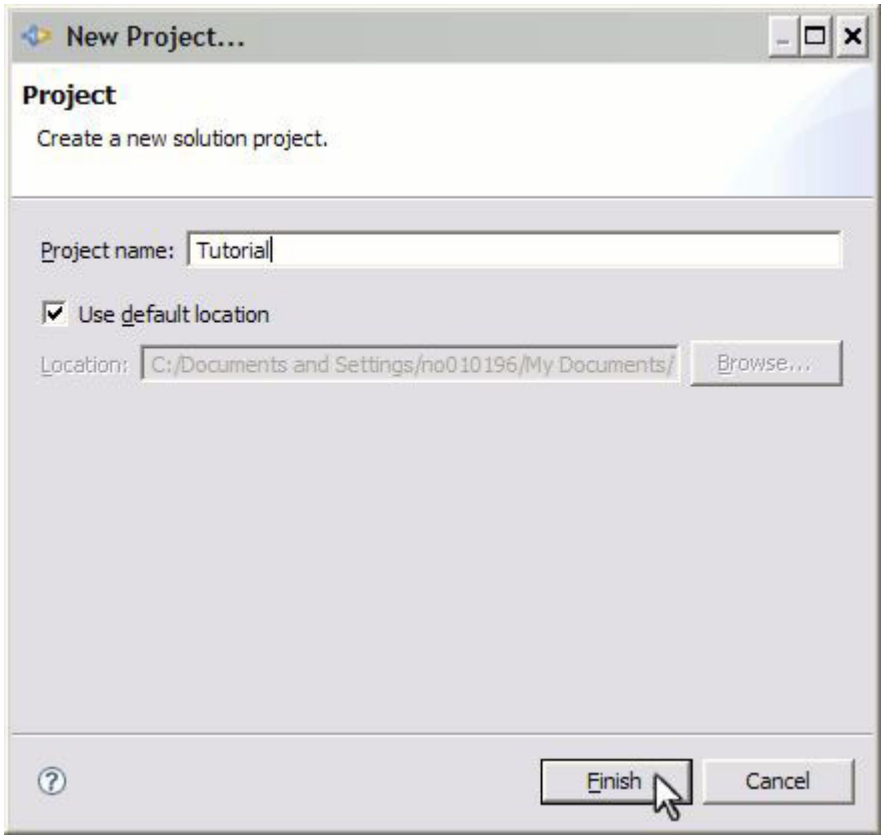

図 7. 新規プロジェクトの命名

主な開発ワークエリアが表示されます。ここにあるパネルはすべてサイズ変更が可能で、画面の編成を決めることができます。ここで画面に表示されているのが「IBM Security Directory Integrator」のデフォルト・パースペクティブです[8]。

8. パースペクティブ とは、単に開発環境パネルの編成を意味します。レイアウトを変更してデフォルトの「IBM Security Directory Integrator」パースペクティブに戻す場合は、一番上のメニューの「**ウィンドウ**」をクリックして、「**パースペクティブのリセット**」オプションを選択してください。

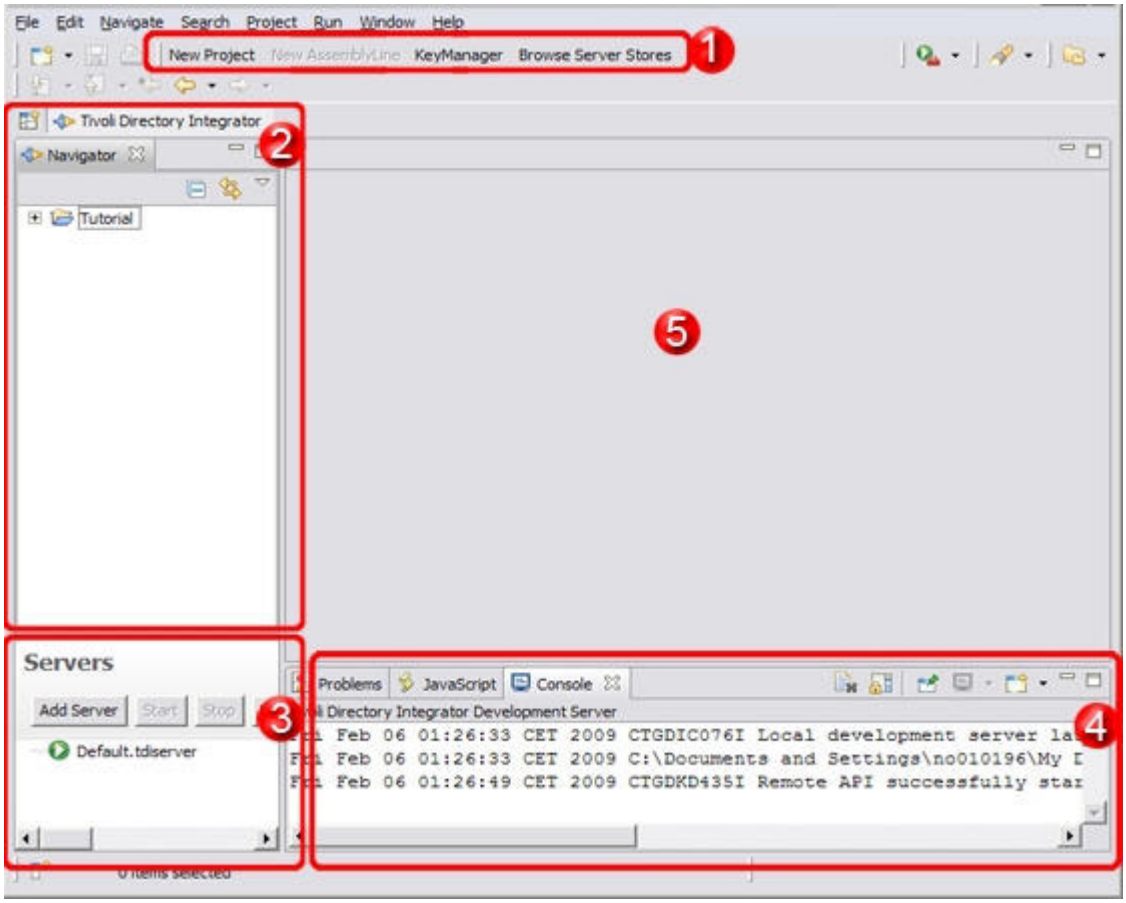

図 8. 構成エディターのメインスクリーン

注: 使用する必要があるパースペクティブは、**Security Directory Integrator** パースペクティブまたは **Easy ETL** パースペクティブのみです。構成エディターが希望の表示になっていない場合や必要なペインが表示されていない場合は、以下の手順を実行してみてください。

- **「ウィンドウ」 > 「パースペクティブのオープン」 > 「Security Directory Integrator」**を選択します。このオプションでは Security Directory Integrator パースペクティブが選択されます。
- それでも解決しない場合は、**「ウィンドウ」 > 「パースペクティブのリセット ...」**を選択します。このコマンドにより、すべてのデフォルト・ペインが強制的に所定の位置に配置されます。

これが、IBM Security Directory Integrator で作業するときに最もよく使用するメインスクリーンです。ここではすべてのナビゲーション・エレメントの詳細に触れないで[9]、上のスクリーン・ショットで強調表示されている番号を見ていきましょう。

1. メイン・ボタン行の中に、新規プロジェクトを作成するためのショートカット一式があります。ナビゲーターでプロジェクトが選択されると、新規 AssemblyLine 作成用のボタンが表示されます。また、証明書キーおよびトラストストアを扱う KeyManager ツールを起動するボタン、およびプロジェクトに関

9. たいていの Eclipse ベースのアプリケーションでは、同様の操作を複数の方法で実行できます。*Directory Integrator の構成* では、使用可能な各種オプションおよびパネルのすべてが説明されています。

連付けられた IBM Security Directory Integrator サーバーからさまざまなプロパティー設定を取得する「**サーバー・ストアのブラウズ**」ボタンもあります。

2. これがナビゲーター・パネルです。ここに開発資産のツリー・ビューが表示されます。新規の「チュートリアル」プロジェクトは、ここに表示されます。
3. サーバー・パネルには、構成されたすべてのサーバーの状況が表示されます。「Default.tdiserver」の隣にある矢印のアイコンは、このサーバーが開始されたことを示しています。このパネルには、新規サーバーの定義、サーバーの開始と停止、リストのリフレッシュ、およびサーバーのログ表示のボタンもあります[10]。

   ここで留意することは、 AssemblyLine を起動するたびに、構成インスタンス[11]と AL の両方がこのパネルに表示される、ということです。
4. ここにはタブ一式が表示されており、現在選択されているタブがサーバーのコンソール出力を表示しています。ここに表示されているメッセージは、サーバーが実行中であり、その API が初期化されて準備が完了していることを示しています。
5. このスクリーン・ショットの灰色の領域には、AssemblyLine およびコンポーネントを作成して開くことにより、 エディター ・パネルが表示されます。リソースのタイプ (コネクター、パーサー、AssemblyLine など) ごとに、専用に作成されたエディターがあります。

最初の AssemblyLine の作成

概要で示した実例シナリオに戻り、 D2 からのデータの結合は当面は無視して、D1 から D3 に情報を移行させる AL を作成します。

図 9. 2 つのデータ・ソースに単純化したシナリオ・ダイアグラム

「チュートリアル」フォルダー (*TDI インストール・ディレクトリー* `/examples` をソリューション・ディレクトリーにコピーしておく必要があります) には、`People.csv` というファイルが格納されています。

```
First;Last;Title
Bill;Sanderman;Chief Scientist
Mick;Kamerun;CEO
Jill;Vox;CTO
```

10. 何らかの理由でサーバーが正しく開始されなかった場合は、「TDI サーバー」を開いて「Default.tdiserver」をダブルクリックします。これにより関連するサーバー文書が開きます。インストールおよびソリューション・ディレクトリーの設定が正しいことを確認してから、このパネルの上部にある「**ソリューション・ディレクトリーの作成**」オプションをクリックしてください。それでも問題が解決しない場合は、サポート担当員にお問い合わせください。

11. IBM Security Directory Integrator サーバーが構成をロードするたびに、そのプロジェクトの AssemblyLine をカプセル化する構成インスタンスを作成し、それら自身が含まれている環境での実行を許可します。同じサーバーに同じ構成を複数回ロードすることができるため、同じ AL のセットを含んだ別々の構成インスタンスが互いに干渉することがない、ということを意味します。

```
Roger
Gregory;Highpeak;VP Product Development
Ernie;Hazzle;Chief Evangelist
Peter;Belamy;Business Support Manager
```

上記のリストから、これがコンマ区切り形式 (CSV) であることがわかります。このファイルは、D1 の入力データ・ソースを表しています。 AL はこのデータを抽出して D3 の出力ターゲットとなる XML 文書に変換します。

ツールバーの最上段の「**新規 AssemblyLine**」をクリックして、新規 AL「CSV2XML」を呼び出します。

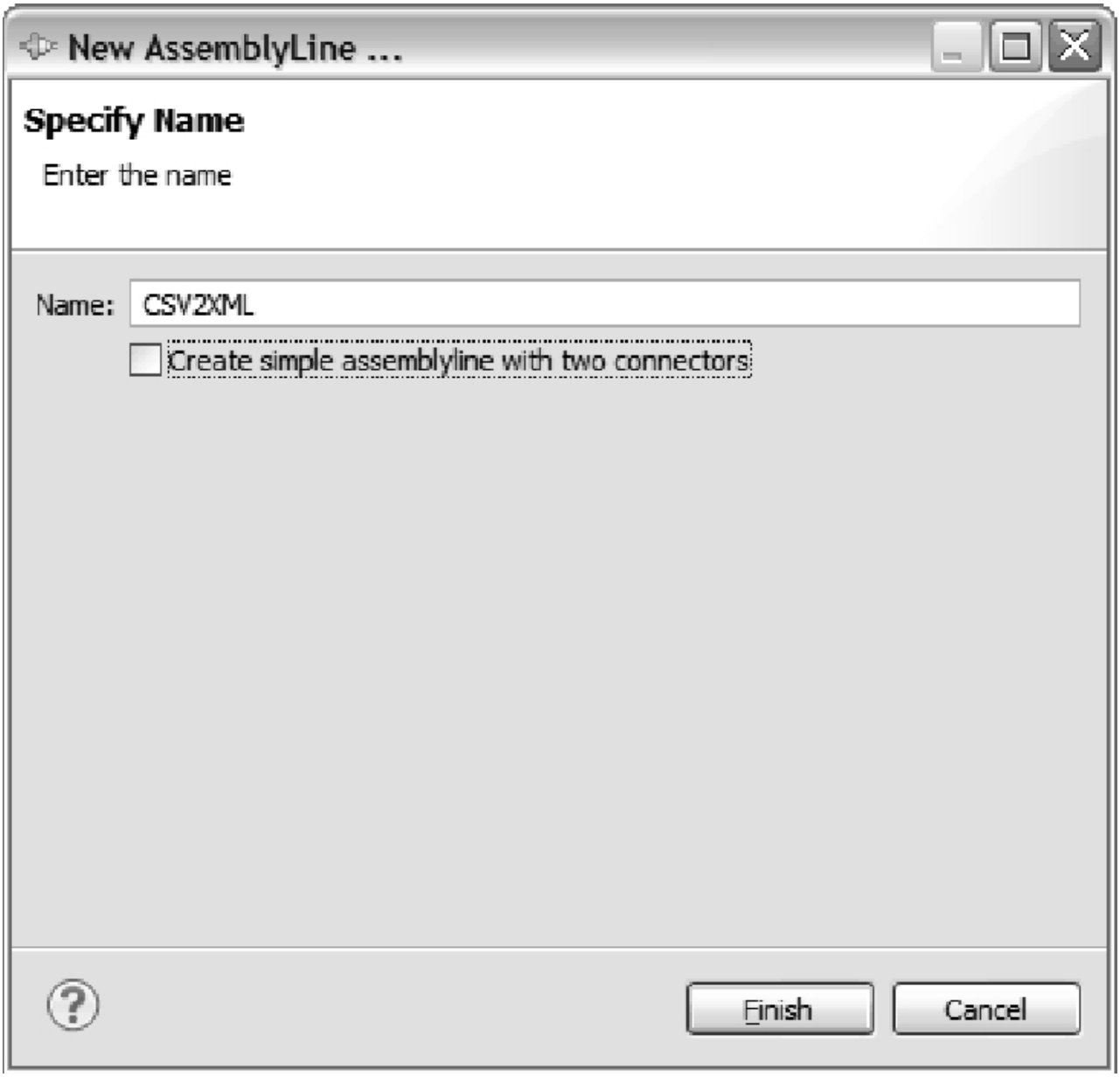

図 10. 新規の AssemblyLine ダイアログ・ボックス

「終了」ボタンをクリックして、AssemblyLine エディター・タブの AL をオープンします。

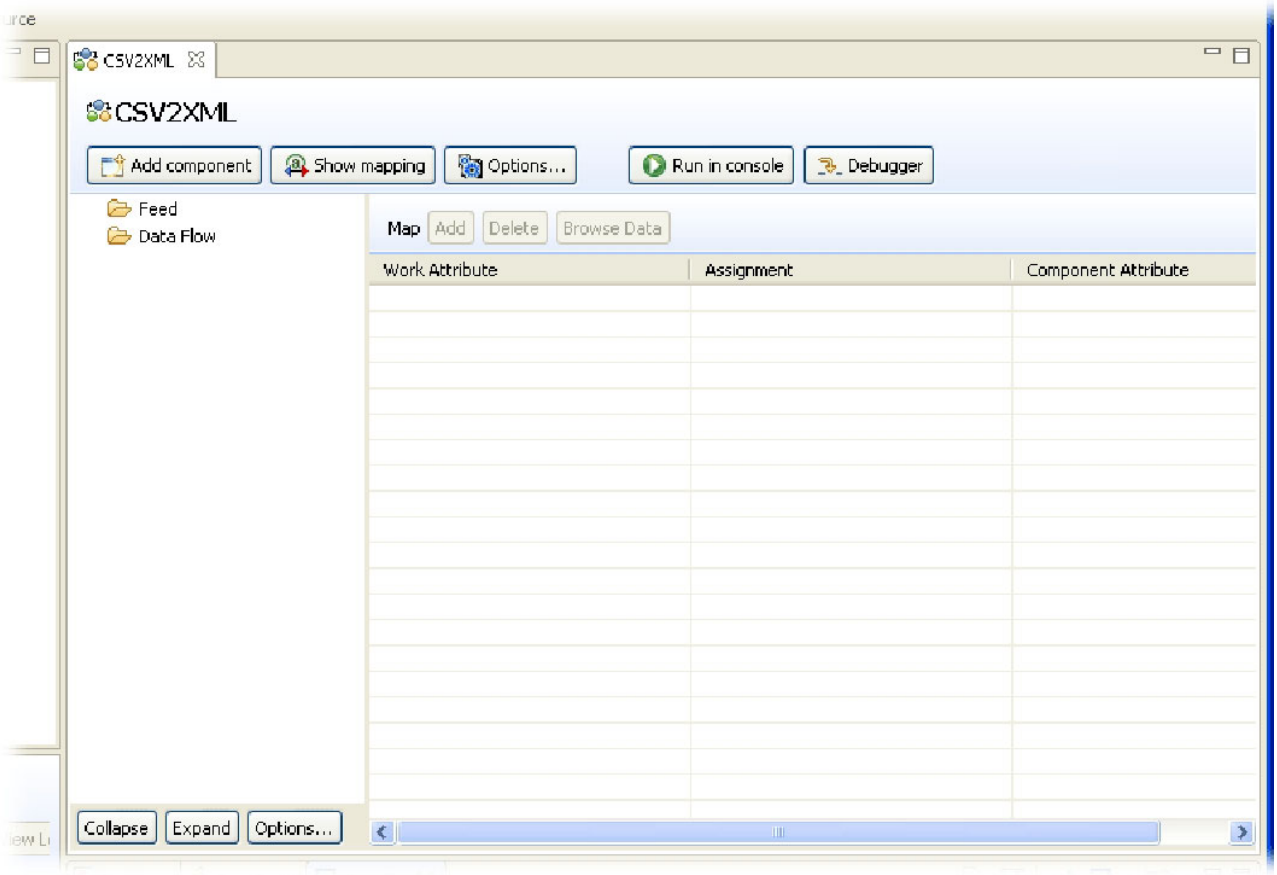

図 11. 空の AssemblyLine エディター

AL エディターの左側にはこの AssemblyLine を構成しているコンポーネント・リストが含まれており、セクション名の「フィード」と「データ・フロー」以外はまだ空になっています。右側の領域には AL にマッピングしたり、マッピングからはずしたりする、すべての属性が表示されます。

これらの AssemblyLine セクションを理解するため、この新規の AL で何が可能になるか少し考えてみます。*CSV ファイルの各行で、XML 文書中に新規のノードを作成します。*このループする動作は IBM Security Directory Integrator カーネルによって自動的に提供されますが、フィード・セクションのコネクターからの入力データが存在する限り、AL のデータ・フロー・セクションの下部にリストされているコンポーネントを操作します[12]

CSV 入力ファイルを読み出すフィード・セクションにコネクターを追加してこの機能を活用しましょう。これを行うには、「フィード」セクション・フォルダーを右

12. 一度に、AL にデータを配布するフィード・コネクターは 1 つだけであることに注意してください。ここで複数のイテレーター・コネクターを配置すると、ライン上の次のコネクターがソースからデータの読み出しを始めるまで最上位のコネクターは当初は空になっています。

クリックしてから、「**コンポーネントの追加...**」を選択します。

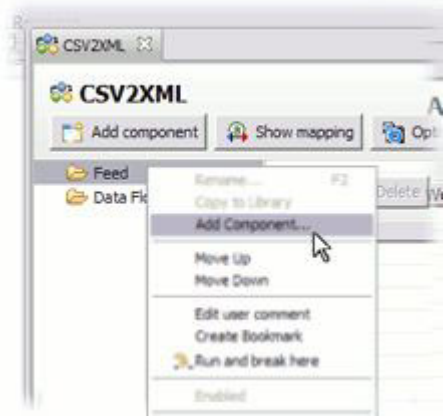

図 12. 新規コンポーネントの挿入

「**コンポーネント選択**」ウィザードが表示されます。

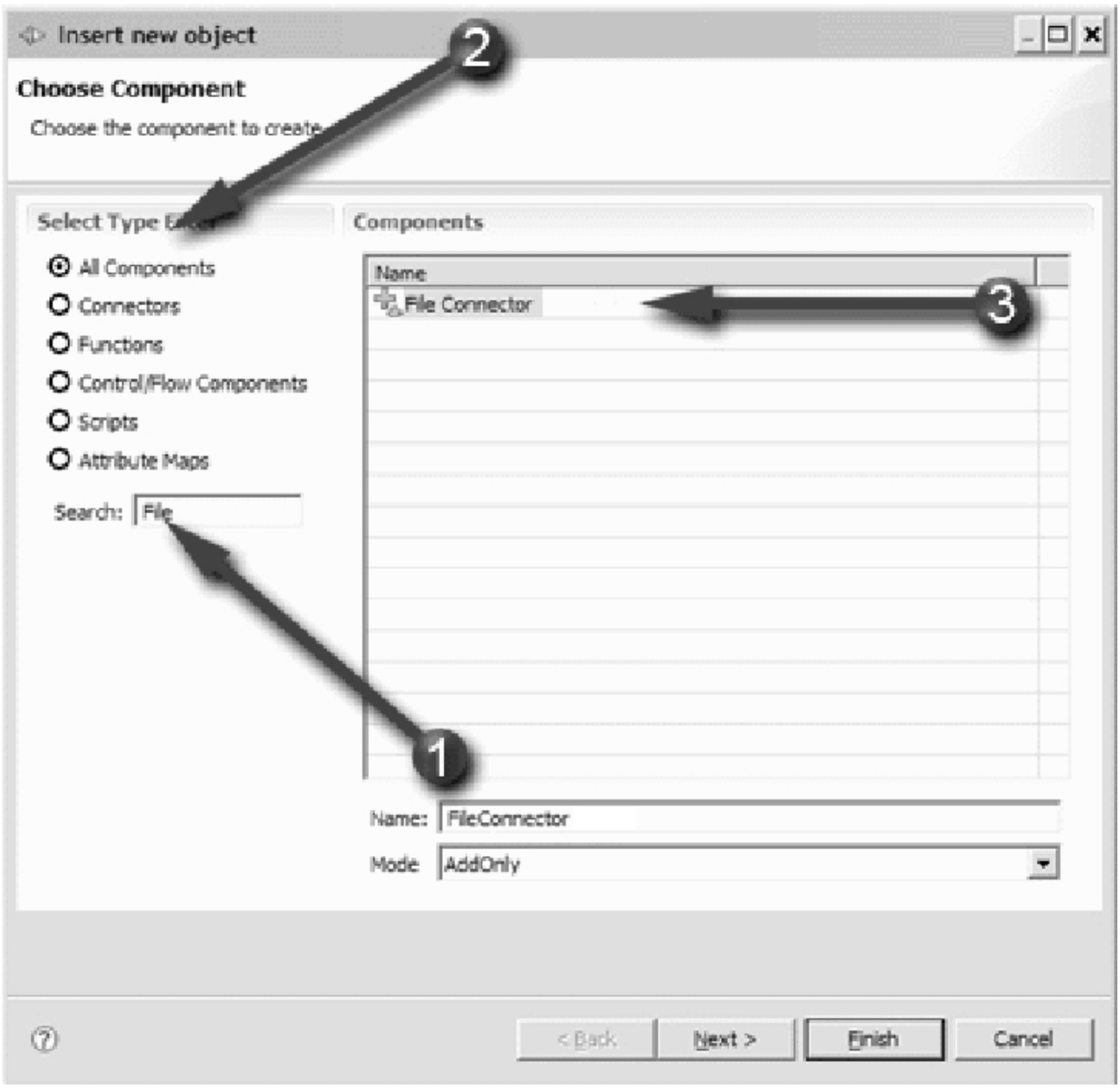

図 13. コンポーネントの選択

このダイアログは、必要なコンポーネントを検索して選択するためのいくつかのオプションを提供します。

1. テキスト・フィールドにコンポーネント名の任意の部分を入力し始めると、入力に従って右側に選択リストがフィルタリングされます。この例では「ファイル」と入力しています。
2. オプションとして、コネクター、パーサー、スクリプト等といったコンポーネントの 1 つのタイプのみを含めることで、選択リストに制約をかけることができます。
3. このリストから必要なコンポーネントを見つけて選択します。この例では「ファイル・コネクター」を選択します。

新規コネクターは自動的に「ファイル・システム・コネクター」と命名されています。「Read_CSV_File」に名称変更すればよりソリューションの文脈に沿ったものになりますので[13]「**モード**」ドロップダウン・リストから「**イテレーター**」を選択します。

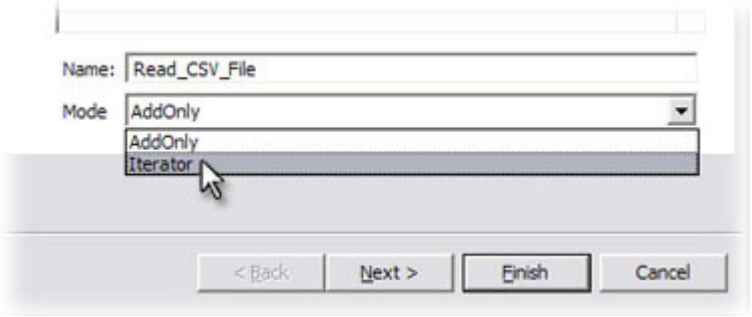

図 14. コネクターのリネームとモード変更

フローにおけるこのコンポーネントの役割を、組み込みの AL 実行ロジックに通知するのは、コネクターのモード設定です。 CSV ファイルから一度に 1 項目のデータを引き出して、データ・フロー・セクションに追加するコンポーネントに格納するために必要となる「*for-each* 動作」がイテレーター・モードです。

「**次へ**」ボタンをクリックして選択されたコネクターの構成パネルに続きます。

13. コネクターには任意の名を付けることができますが、先頭が文字で始まり、任意の文字、数字、下線文字が後に続くスクリプト変数と同じ命名方式にすることを推奨します。これは、後で簡単にスクリプト・コードから再構成して駆動させられるように、すべての AL コンポーネントがスクリプト変数として自動的に登録されるためです。

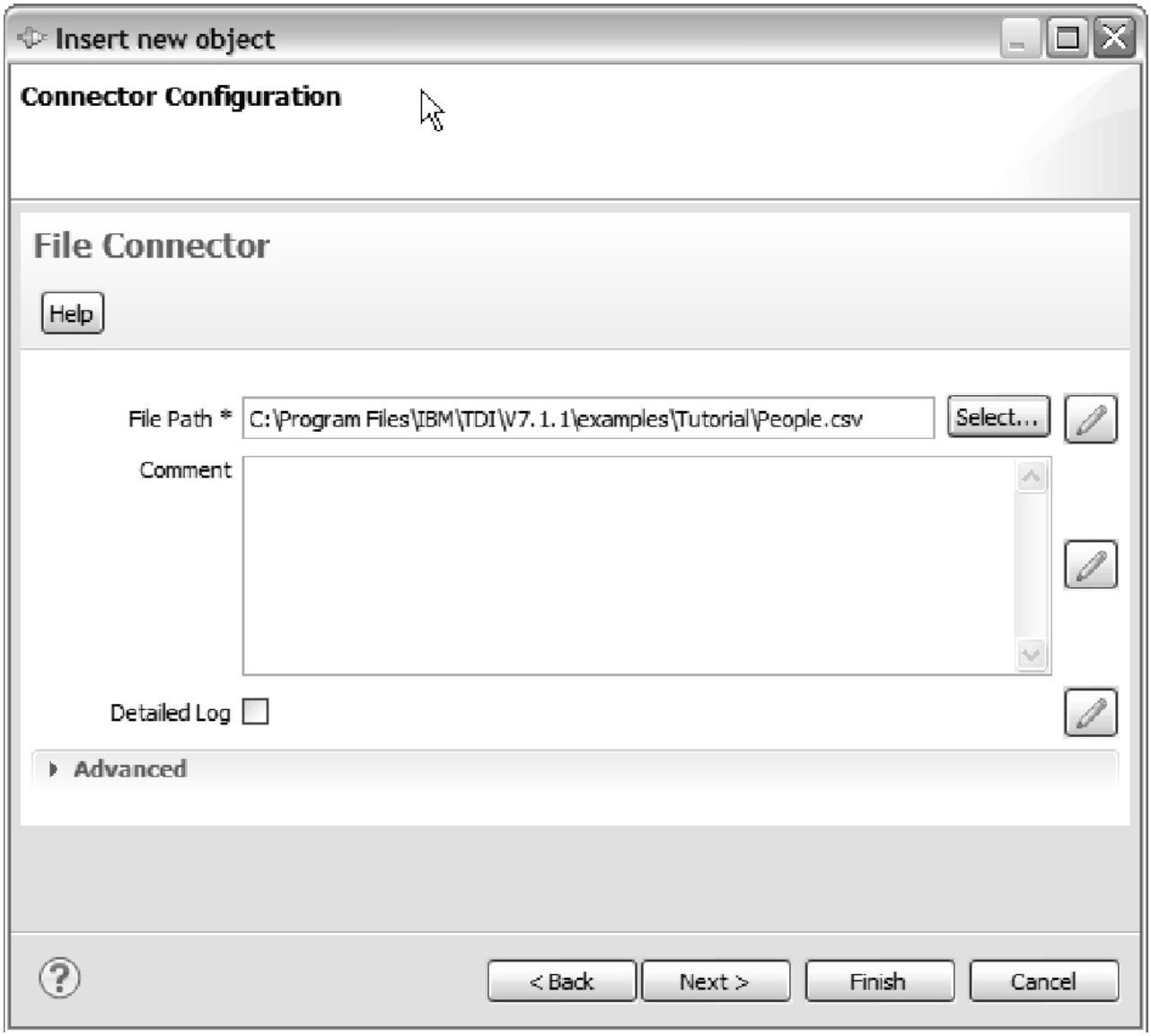

図 15. ファイル・コネクター構成パネル

各コンポーネントは固有の構成パラメーター・セットを提供しています。この画面上に表示されているのはファイル・コネクター用のもので、「**ファイル・パス**」という 1 つのみの必須パラメーターがあります。上記のスクリーン・ショットに表示されているように、絶対パス名またはソリューション・ディレクトリーからの相対パス名を `People.csv` ファイルへのパスとして入力します。[14] または、このファイルを選択するファイル・ブラウザーを表示させる「**選択**」ボタンをクリックします。

定形式テキスト・ファイルはバイト・ストリームであって、データベースやディレクトリーのような構造化されたデータ・ソースではないので、読み取りの時にバイト・ストリームのフォーマットを解釈するパーサーをセットアップする必要があり

14. この技法を使うことで、ソリューションの移動および共有が簡単になります。目的のソリューション・ディレクトリーを指定するのみで済み、すべて相対パスが変更なしで機能するためです。

ます。 IBM Security Directory Integratorはコネクター/パーサーの選択と構成の対話式のテスト用の強力で多機能なデータ・ブラウザー機能を提供します。

この後で、データ・ブラウザーを確認しますが、まず「**次へ**」ボタンを再度押して「**パーサー構成**」へと進めて、このウィザードを終了させる必要があります。それを選択するには、ここで「**CSV パーサー**」をクリックします。

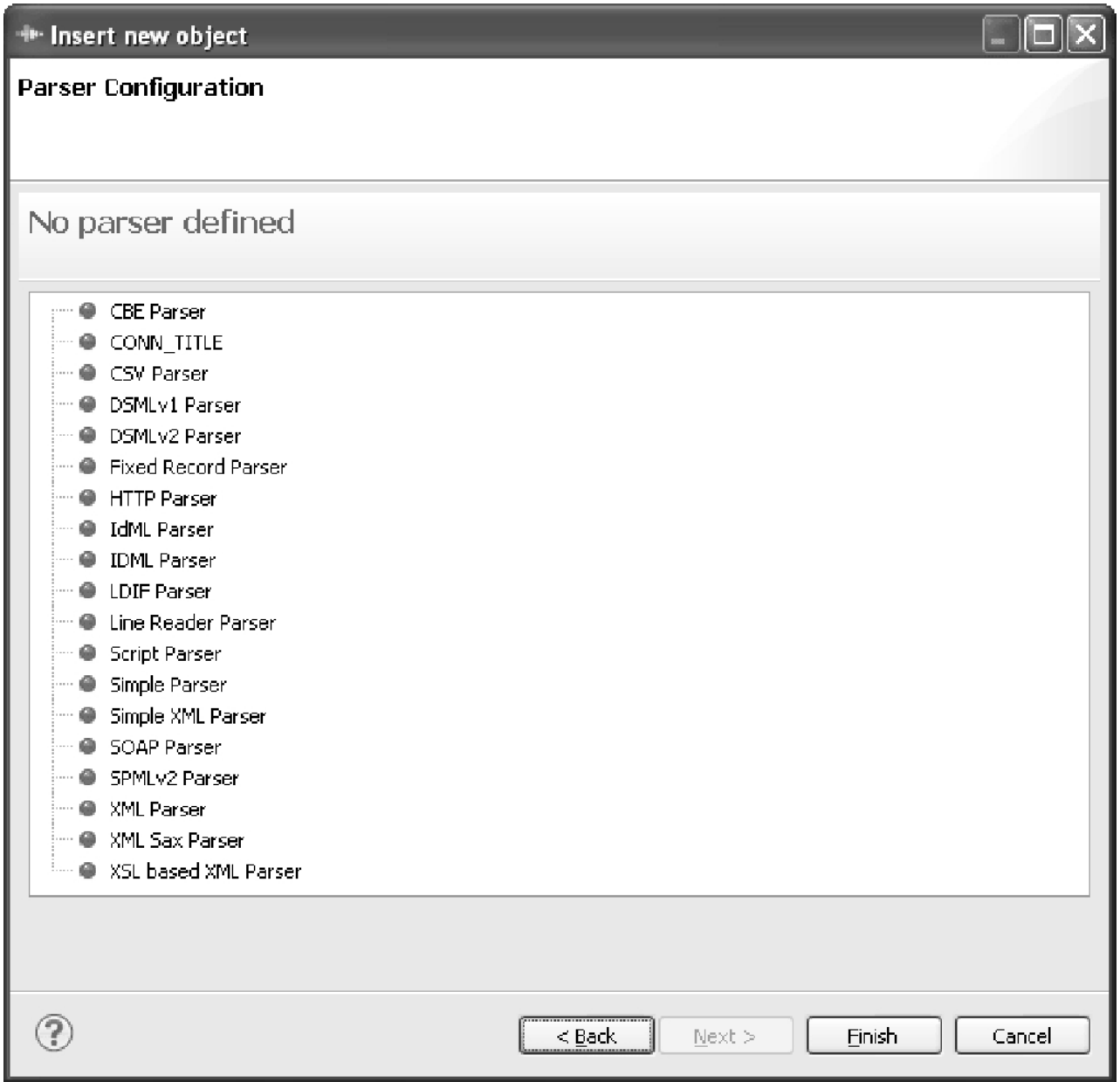

図 16. 新規オブジェクト挿入中のパーサー選択

CSV パーサーを選択したら、「**終了**」を押してウィザードを閉じます。 パーサー構成パネルが表示されます。デフォルト設定を変更する必要がないので、再度終了ボタンをクリックするだけでウィザードを終了します。

次に、その属性値を AssemblyLine にマッピングさせるための入力ソースのスキーマを、コネクターを使って*検出* します。ここで、データ・ブラウザーが役に立ちま

す[15]。AssemblyLine コンポーネント・ツリーの新規のイテレーター・コネクターを右クリックし、コンテキスト・メニューから「**データのブラウズ**」を選択して、データ・ブラウザーを開始します。

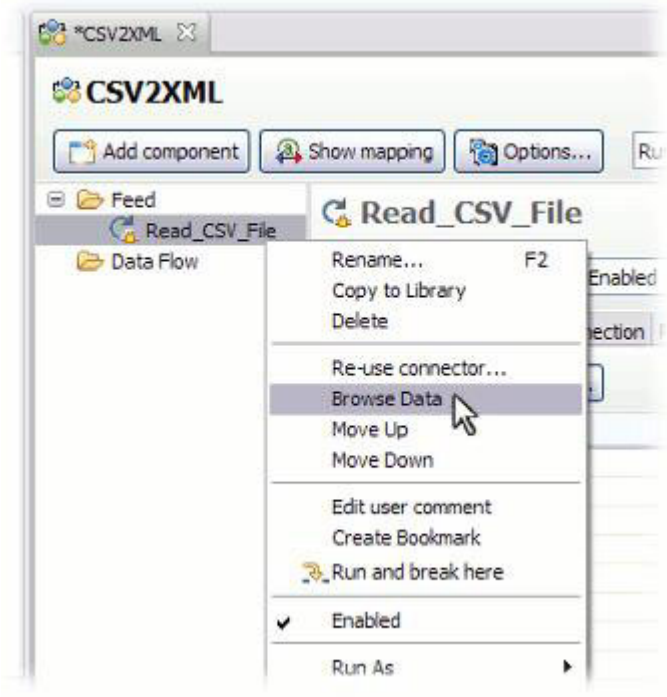

図 17. 「データのブラウズ」コンテキスト・メニューの選択

新規のエディター・タブ中のデータ・ブラウザーをオープンします。

15. ファイルが CSV 形式とわかっているので、イテレーター・コネクターのスキーマ領域の「**接続**」と「**次へ**」ボタンをクリックするのが最短の方法です。次に、必要に応じて入力マップにディスカバーされた属性をドラッグします。形式が不明である場合には、データ・ブラウザーが有効です。それでも、試してみる価値はあります

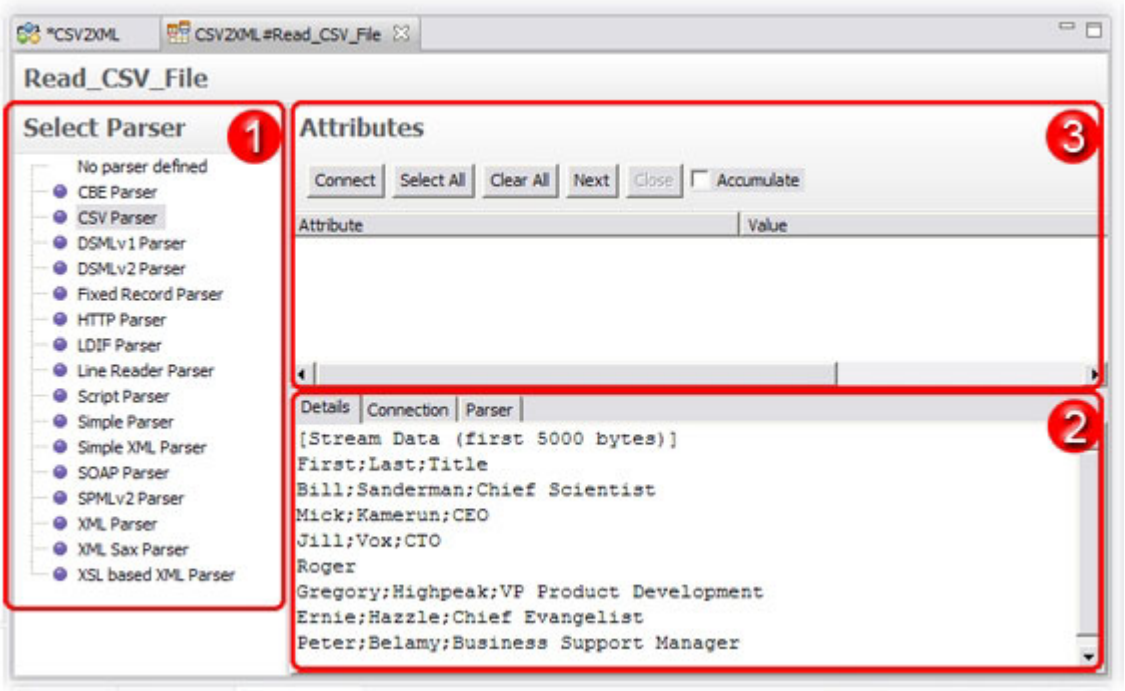

図 18. データ・ブラウザー

上記のスクリーン・ショット中のラベル 1 とされている領域は、選択済みパーサーの指定と変更を行います。領域 2 は、構文解析される未加工のバイト・ストリームの内容をそのまま表示する詳細タブです。選択されたパーサーを構成するタブと共に、コネクターの接続パラメーター変更用のタブもあります。

このダイアログの最終セクション (スクリーン・ショットの 3 の領域) では、データ・ソースへの接続と使用できる属性の検出を行います。初めに「**接続**」ボタンを、次に「**次へ**」ボタンをクリックします。

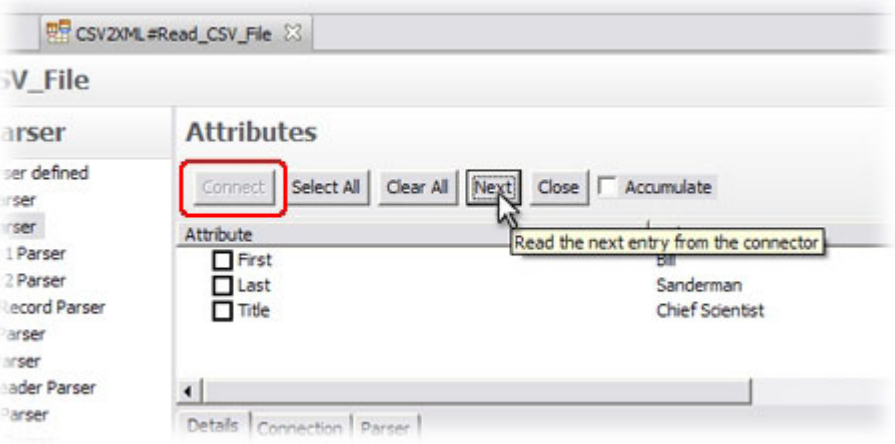

図 19. 実際にデータを見ながら、対話的にスキーマを検出する

このファイルのスキーマが検出されました。各属性項目の隣にあるチェック・ボックスを選択するか「**すべて選択**」ボタンを使って、この場合はすべてとなっている、マッピングしたい属性を選択します。

Ctrl+W のショートカットによりデータ・ブラウザー・タブを閉じるか、単純にタブの右端の「クローズ」シンボル (X) をクリックするかして、AL が下方のスクリーン・ショットのようになっている AssemblyLine エディターに戻ります[16]。

図 20. イテレーター・コネクターを配置した AL

選択されたコンポーネントの詳細は、入力マップに設定された 3 個のマッピング規則を含めて AL コンポーネント・リストの右に表示されます。属性マップの個々の項目には「**割り当て**」がありますが、それはターゲット属性の (複数の) 値を設定するために評価されるスクリプトのスニペットとなっています。

処理を続ける前に、これらの*割り当て* について、少し考えてみましょう。 AL にトランスポートされるすべてのデータを伝達する、包括的に有効な*作業項目* を AssemblyLine が有しているとしている 5 ページの『項目属性値データ・モデル』セクションのことを思い出してください。このオブジェクトは、登録済みのスクリプト変数 `work` を使ってスクリプト・コードで参照されます。さらに、すべてのコネクター・インターフェースは読み出しと書き込み用のキャッシュとして使われる固有の *Conn 項目*を持ちます。コンポーネントに固有のこのオブジェクトは、事前登録された変数 `conn` により、スクリプトからアクセスされます[17]。イメージを描くために、最初のマッピング規則について考えてみてください。それは「First」と命名された作業項目の属性を生成します。その値は以下の割り当てによって得られます。

16. 何らかの理由でコネクターがデータ・フロー・セクションにある場合には、フィードにドラッグするだけです。「**モード**」設定が「**イテレーター**」でない場合には、コネクターを右クリックしてモードを選択してからイテレーターを選択します。

17. IBM Security Directory Integrator フック・フロー・ダイアグラムに表示されているように、`conn` 変数は一定の期間のみ有効です。このスコープの範囲外では、その conn 項目に対応するコンポーネントを照会することでアクセス可能です。

```
conn.First
```

この省略表現は conn 項目に読み込まれたばかりの「First」と呼ばれる属性を参照したものであり、その値は新規の作業項目の属性を設定するために使われます。対応する割り当てのスクリプトは以下のようになります。

```
return conn.getAttribute("First");
```[18]

練習に戻って、ここでターゲットの XML 文書 (データ・ソース D3) 作成用の出力コネクターを追加する必要があります。ここでは、AssemblyLine コンポーネント・パネルの最上部の「**コンポーネントの追加**」ボタンを使います。

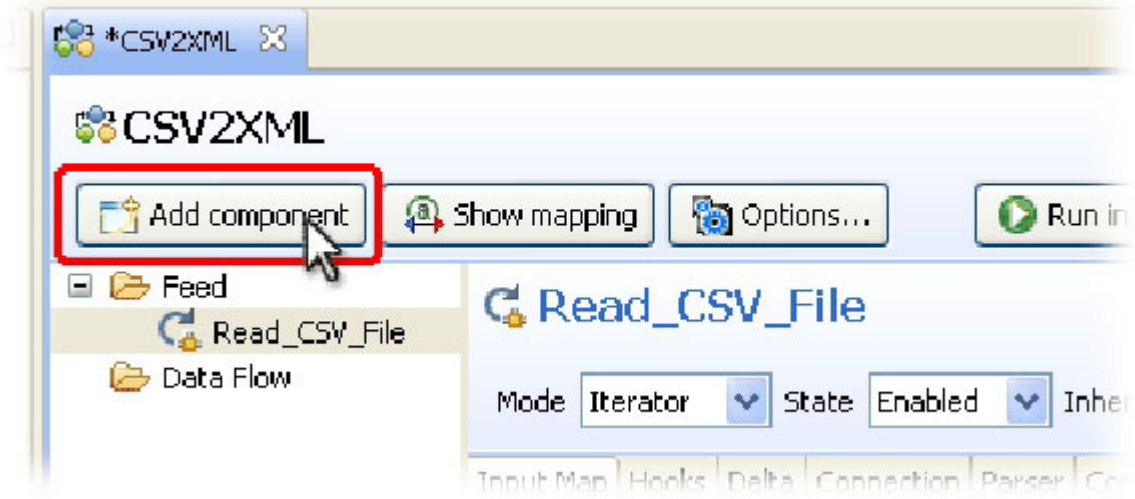

図 21. コンポーネント・ボタンの追加

再度、ファイル・コネクターを選択し、「Write_XML_File」に名前を変更します。モード設定を「**AddOnly**」のままにして「**次へ**」ボタンをクリックします。

コネクター構成パネルで、チュートリアル・フォルダーに Output.xml というファイルを書き込むための「**ファイル・パス**」のパラメーターを設定します。次のウィザード・パネルで「XML Parser」を選択します。 XML パーサーの構成を変更する必要がなくなれば、「**終了**」ボタンをクリックできます。出力コネクターの場合には、検出する Output.xml ファイルがないためスキーマ探索を使用できないことに注意してください。

18. バージョン 6.x 以前からのユーザーであれば、バージョン 7.0 より以前の構文を使うこともできます。

```
ret.value = conn.getAttribute("First")
```

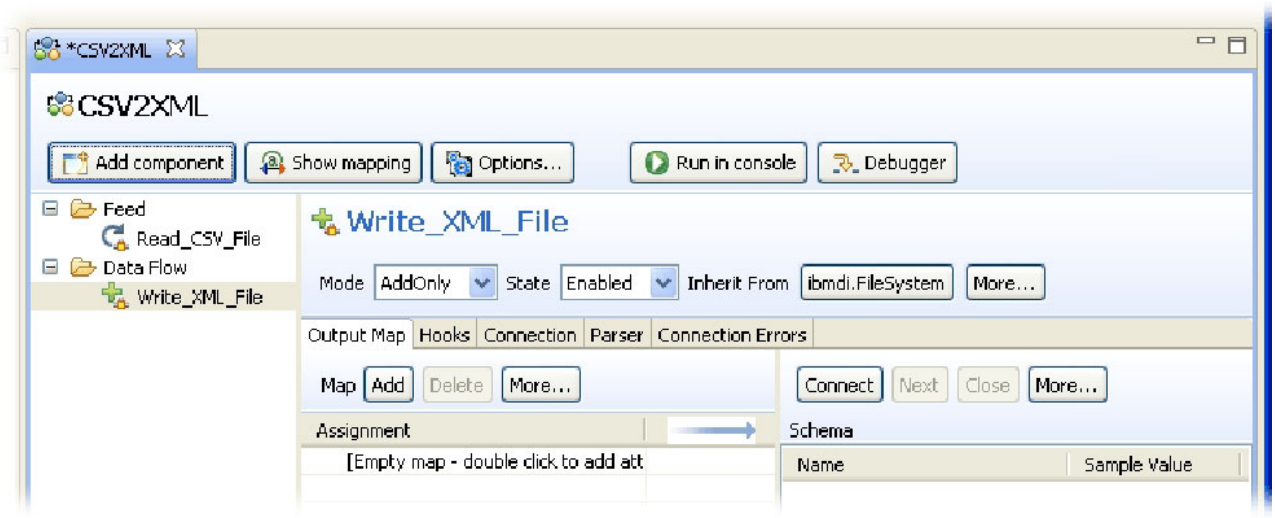

図 22. 2 つのコネクターを配置した AL

コンポーネントを選択すると、エディター画面の右側にその詳細が表示されるのがわかります。「フィード」または「データ・フロー」フォルダーを選択すると、この AssemblyLine 用のすべての「**属性マップ**」の概要が表示されます。この表示は、入力属性を最新のコネクターの出力マップにコピーするには便利なので、「フィード」か「データ・フロー」をクリックしてこの画面を表示させます。

イテレーター・モード・コネクターによって AL に取り込まれている (全部で 3 つの)属性リストが表示されます。これらの入力マップ属性を選択します。[19] データ・フローを完了させている「Write_XML_File」コネクターの出力マップまでドラッグします。

19. 複数行を選択するには、CTRL キーを押しながら 1 行ずつクリックしていく、または Shift キーを押しながら範囲の先頭行と最終行をクリックします。

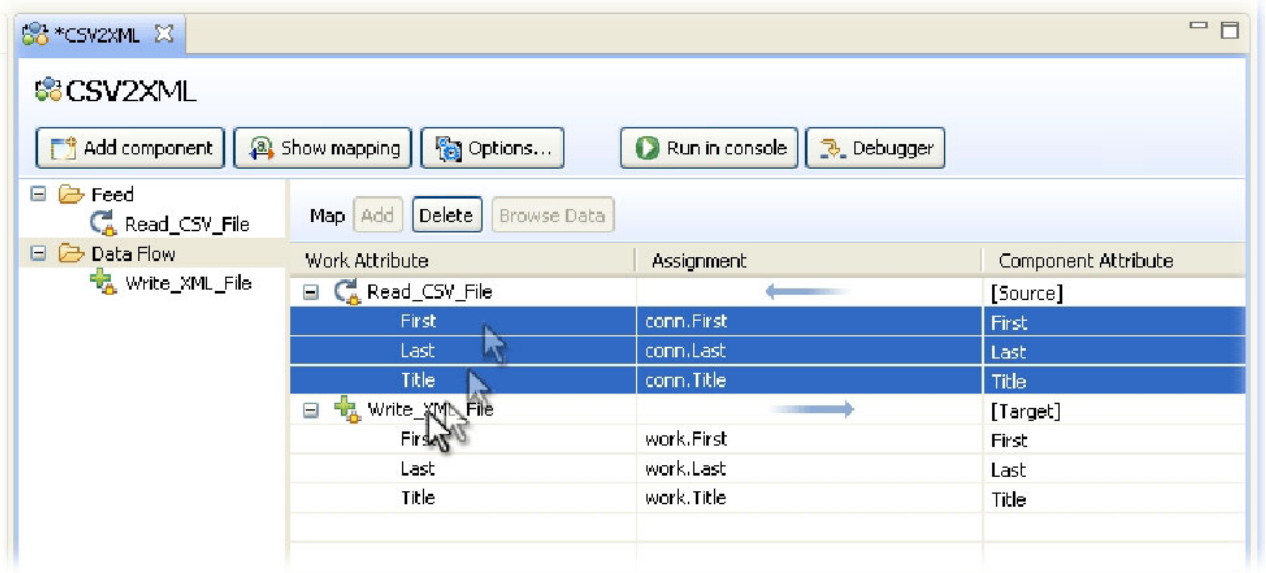

図 23. 出力マップに属性をドラッグする

割り当てが入力形式から出力形式に自動変換される仕組みに着目してください。例えば、「Read_CSV_File」コネクターの入力マップ中の最初の項目は、`conn.First` (Conn 項目に読み込まれた「First」と命名された属性) 中に検出されるすべての値を保持する「First」という作業項目の属性を作成します。この入力マッピング規則を出力マップにドラッグすると、その割り当ては変更されて、値が逆に作業項目から取り込まれ、コネクターのキャッシュ (Conn 項目) 中のターゲット属性を作成します。

マッピング規則用のソースを変更する場合には、その割り当てを編集します。マッピングされている属性名を変更するためには、右クリックして名前を変更するだけです。最初の 2 つの出力マップ規則にこれを実行します。

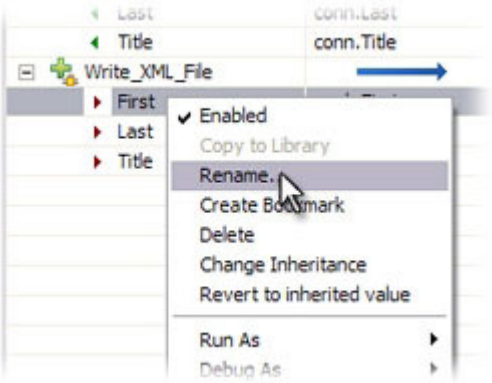

図 24. 属性マップ規則の名前変更

「First」を「FirstName」に、「Last」を「LastName」にします。

「Write_XML_File」出力マップを右クリックし、**「属性の追加」**を選択して、この出力マップに新規のマップ項目を追加します。

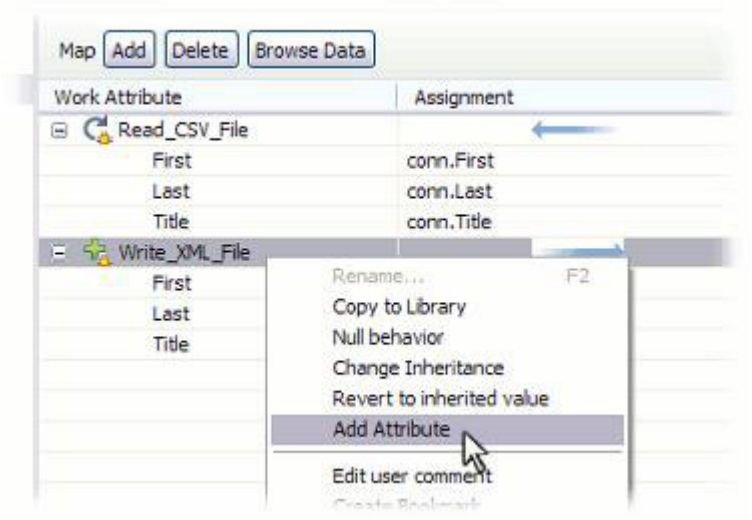

図 25. 出力マップに「FullName」属性を追加

この新規のマッピング規則ターゲットを「FullName」の名前をつけて、「**OK**」ボタンを押してから、ダブルクリックしてその割り当てを編集します。これでスクリプト・エディター・パネルがオープンされ、`work.FullName` というデフォルトの割り当てスクリプトが表示されます。もちろん、作業項目に「FullName」属性はないので、このマップではどのような値も設定することはできません。その代わりに、スクリプトを変更してこの値を計算して、間にシングル・スペースを入れた状態で、First と Last の属性を連結させる必要があります。

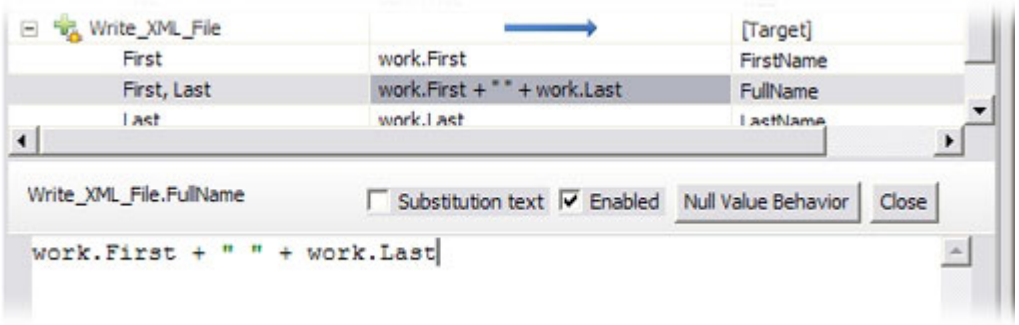

図 26. 割り当ての編集

スクリプトの記述は以下のようになります。

```
work.First + " " + work.Last
```

このような 1 行の属性マップの割り当てスクリプトには終端のセミコロンは不要であることに注意してください[20]。完了したら、スクリプト・エディター・パネルの右上の「**クローズ**」ボタンを押します。

20. バージョン 7.0 より以前の構文もサポートされているので、マップの割り当てスクリプトは依然として "ret.value =" で開始することができます。

Assemblyline の実行

ここで AL をテストしてみましょう。

AssemblyLine エディターの上部にある「**実行**」ボタンを押して、テストを行います。

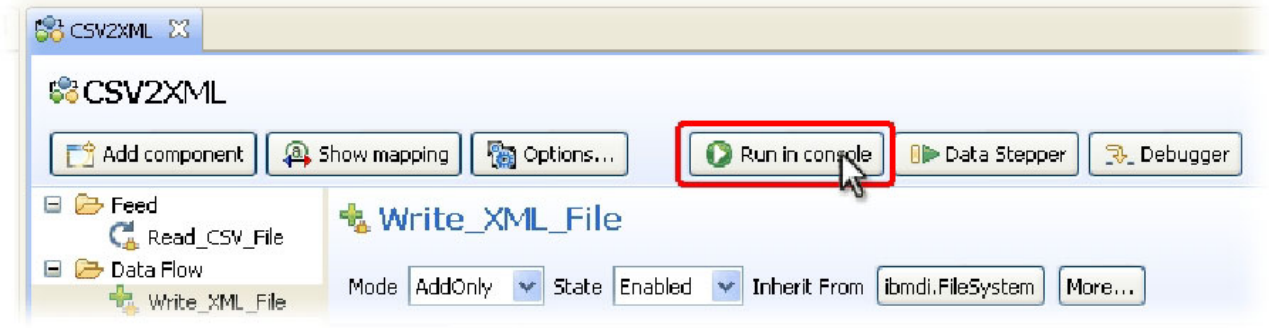

図 27. 実行ボタン

新規のタブが開き、AssemblyLine からのログ出力を表示する*実行ウィンドウ* が表示されます。

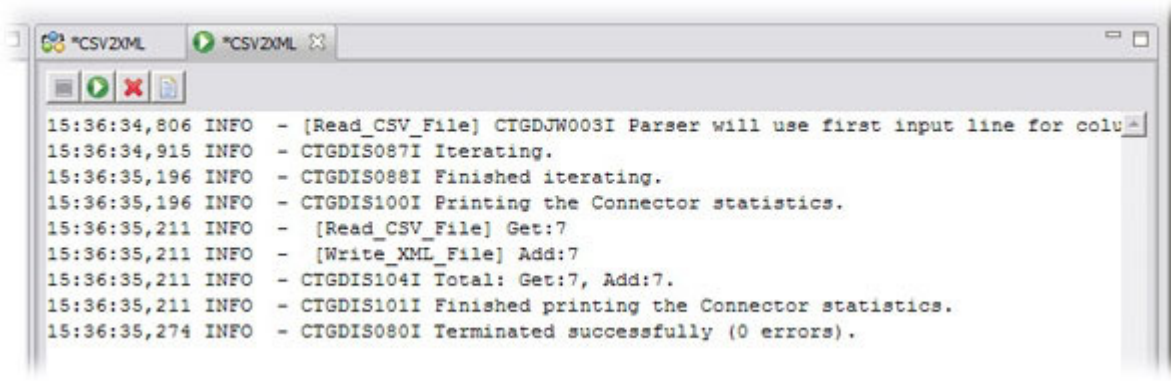

図 28. AssemblyLine 実行時のログ出力

CE は、AssemblyLine をそのすべてのコンポーネントとともに実際に取り出して、*構成*、つまり、IBM Security Directory Integrator サーバーに割り当てる作業を定義する XML 文書をエクスポートしました。この構成をサーバーにパイプ接続して AL を実行する指示を出し、スクリーン内に表示されるすべてのログ出力を収集します。

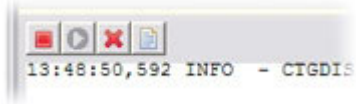

図 29. ログ出力ウィンドウのボタン・バー

実行ウィンドウには、AL の停止、いったん停止した AL の再開、およびログ内容の消去のためのボタン・バーが付いています。右端のボタンは、現在のログ出力を外部エディターで開くためのものです。

AssemblyLine のログ出力の最後に、関係するすべてのコンポーネント (この場合は 2 つのコネクターのみ) の統計情報が表示されます。上記の情報は、CSV ファイルから 7 つの項目が読み取られ、XML 文書に 7 つのノードが書き込まれたことを示しています。

結果を確認するために、出力ファイルをディスクに置いてブラウザー・ウィンドウで開くことができます。出力コネクター (Write_XML_File) を右クリックして「**データのブラウズ**」を選択し、データ・ブラウザーを利用することもできます。

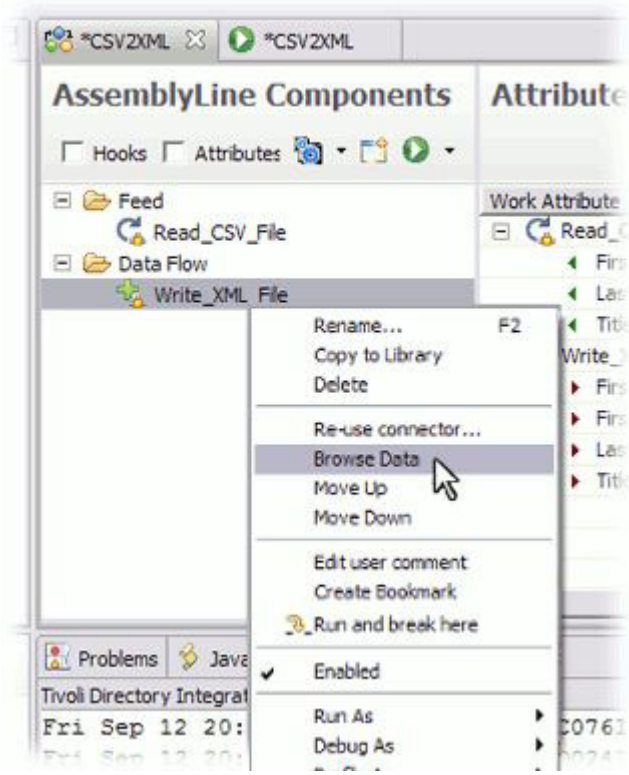

図 30. 出力コネクターで作成されたデータのブラウズ

これにより、選択したコネクターについての構成が終わり、データ・ブラウザーは、実行準備完了状態になります。「**接続**」ボタンを押してから「**次へ**」を押すと、出力データが読み取られて表示されます。

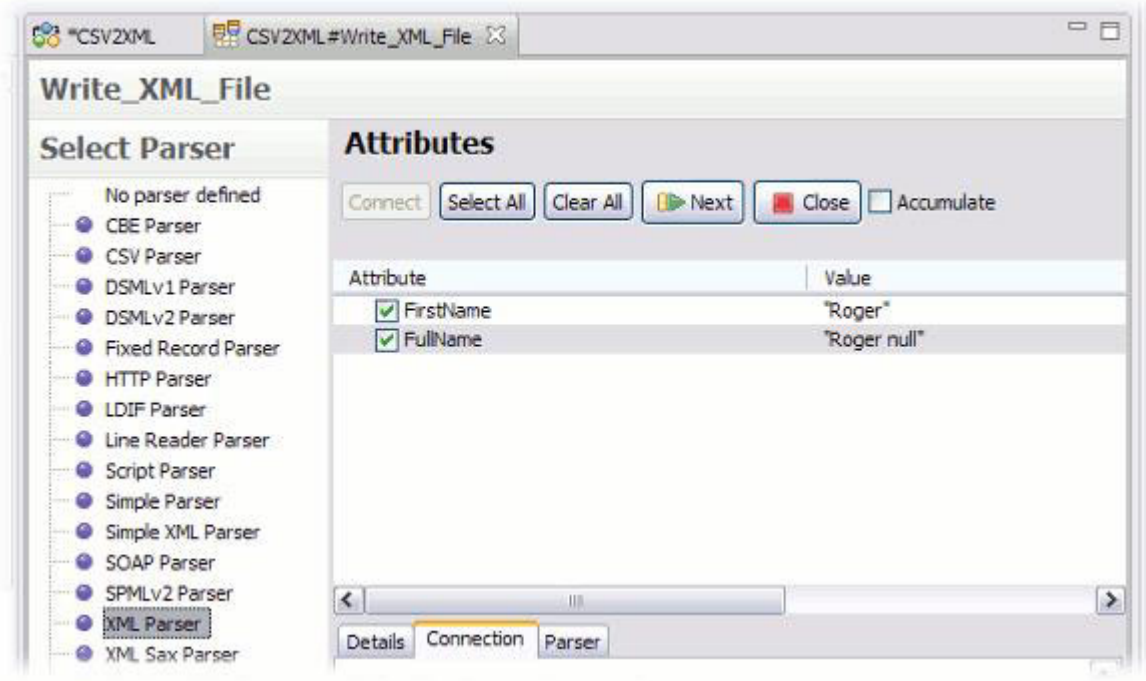

図 31. 出力された XML のブラウズ

出力された XML は、入力された値とマッピングのロジックを正確に表しています。4 番目の項目 (Roger) 以外は問題がないように見えます。4 番目の項目では、「LastName」および「Title」が欠落し、「FullName」の計算値が `Roger null` となっています。

入力データを詳しく調べると、CSV の行の一部が不完全であることがわかります。

```
First;Last;Title
Bill;Sanderman;Chief Scientist
Mick;Kamerun;CEO
Jill;Vox;CTO
Roger
Gregory;Highpeak;VP Product Development
Ernie;Hazzle;Chief Evangelist
Peter;Belamy;Business Support Manager
```

入力データの欠落や間違いはよくあることです。ソリューションでは、これを処理中にフィルターで除去するか、訂正する準備をしてください。

Null 動作: 欠落している属性または値の処理

欠落している値は、属性マップに組み込まれている Null 動作機能を利用するか、自分自身で検出して処理することができます。

では、Null 動作の設定から始めましょう。属性マップ・パネル (AL コンポーネントのツリー・ビューではありません) の「Read_CSV_File」コネクターを右クリック

して、コンテキスト・メニューから「**Null 動作**」を選択します[21]。

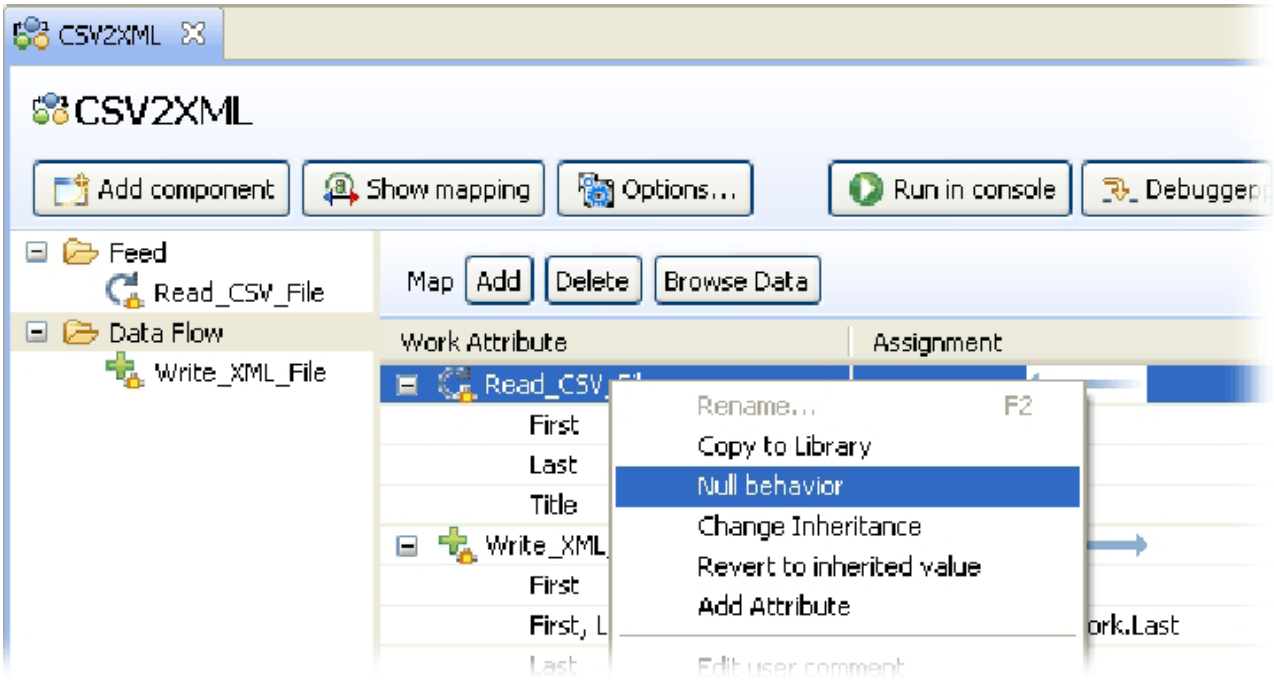

図 32. AL レベル設定の Null 動作ボタン

これで Null 動作のダイアログが立ち上がるので、*null* の定義方法 (読み取るソースのタイプによって変わります) と処理方法の両方を設定することができます。デフォルトでは、*null* は属性が欠落していることを表わします。またデフォルトの処理では、この属性をマッピング操作から除外します。その結果、最終的には指定された名前の属性が受信項目から消去されます。

この場合は、Null 動作ダイアログが開いたら右側に並んだラジオ・ボタンを使用して *null* を空ストリングの値として定義してから、左側に並んだボタンでデフォルト値の "* missing *" が返ってくるように指定します。

21. あるいは、AL コンポーネント・リストからコネクター自体を選択することもできます。入力マップの上部にある「**その他...**」ボタンを押して、「Null 動作」を選択してください。

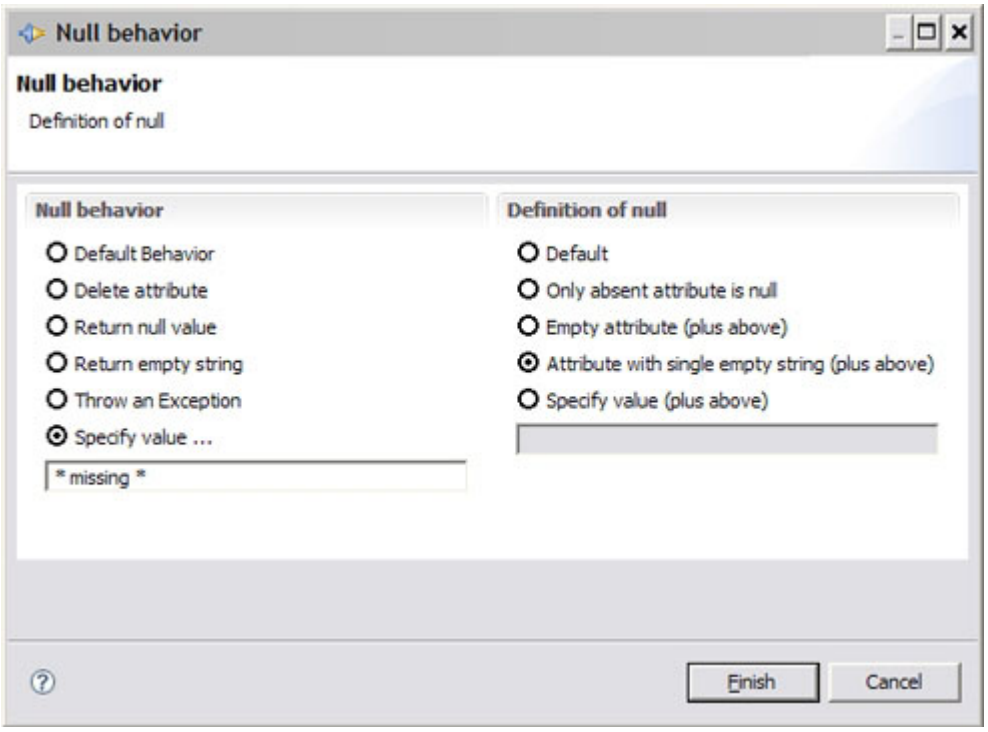

図 33. Null 動作構成ダイアログ

AssemblyLine を再実行して、`Output.xml` の内容が表示されているブラウザー・ウィンドウをリフレッシュします。それで、「Roger」の項目の「Title」および「LastName」には特別な *null* 値 (「* missing *」) が表示されます。

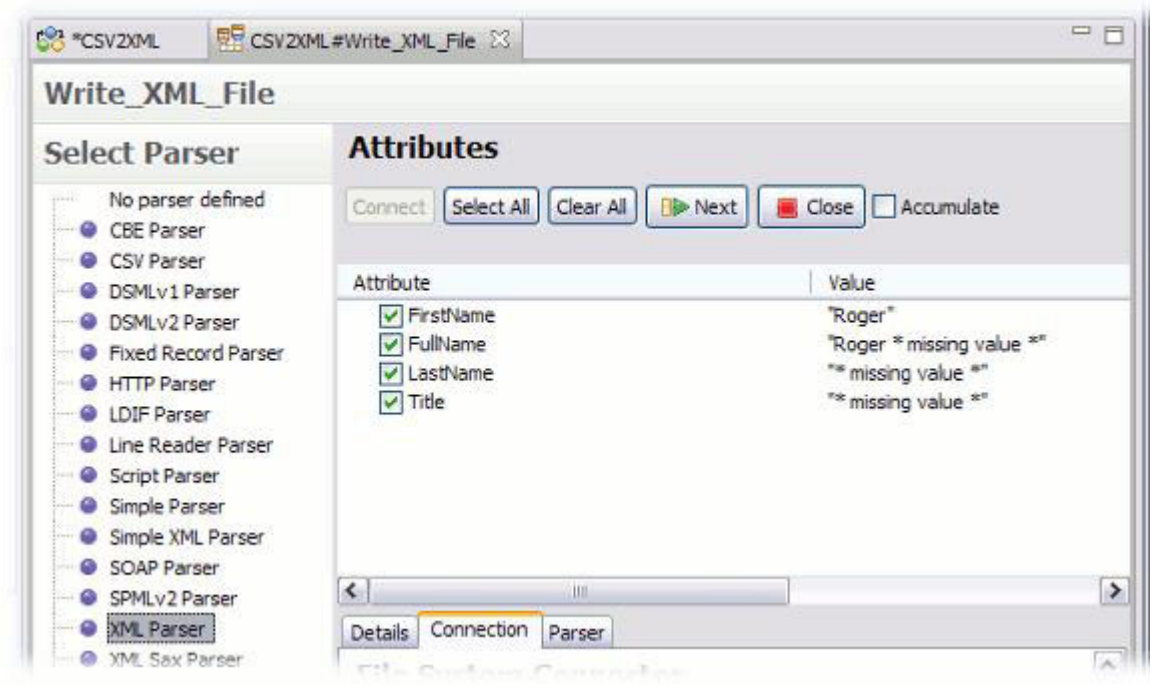

図 34. Null 動作設定の XML 出力の結果

これでいくらか良くなりました。少なくとも、意図的に選択した方法です。しかし、場合によっては、項目が不完全なため処理の続行ができなくなります。今回の練習では、「Fullname」を計算するために「FirstName」および「LastName」の値が最低限必要です。そこで、フィルター処理のロジックを AssemblyLine に追加して、すべての項目がこの要件を確実に満たすようにします。

まず「**コンポーネントの挿入**」ボタンをもう一度クリックして、左側に並んだラジオ・ボタンから「**制御/フロー・コンポーネント**」を選択します。それからリストの「IF」を選択して、「Incomplete data」と名前を付けます[22]。そして「**終了**」を押します。

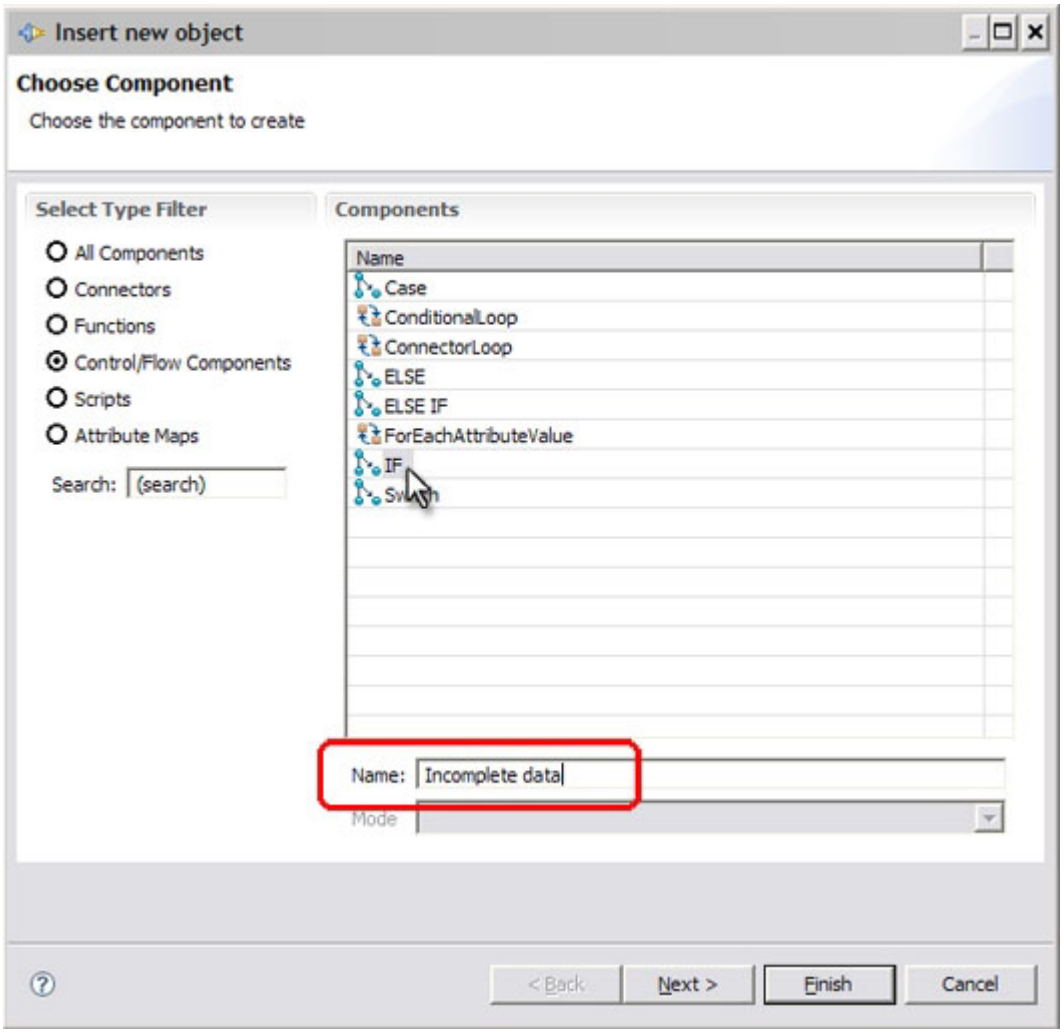

図 35. IF ブランチ・コンポーネントの選択

ここで、この IF ブランチを出力コネクターの上にドラッグします。IF ブランチ・エディターのエリアに条件が追加されます。

22. コネクターにはスクリプト変数と同様に名前をつけることを、本書で先に推奨しました。これは関数コンポーネントにも当てはまります。ただ、属性マップ・コンポーネントや分岐、ループ、スクリプトは、スクリプトから直接アクセスされることがほとんどないため、それほど重要ではありません。本書では、AssemblyLine が見やすくなることを優先しています。

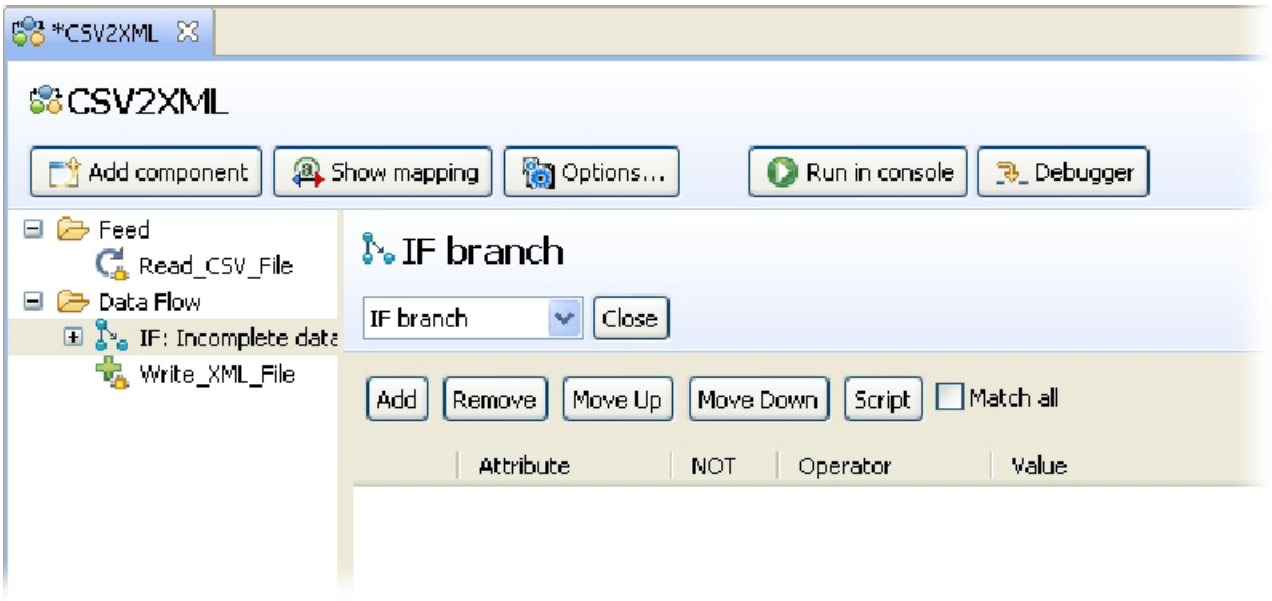

図 36. IF ブランチの条件の編集

ここで、単純条件を追加する方法、またはスクリプトのスニペットを記述する方法、あるいはその両方を選択することができます。「**すべてに一致**」のチェック・ボックスに注意してください。このチェック・ボックスがチェックされていなければ、(単純条件を追加する方法およびスクリプトに記述する方法のいずれも) それぞれの条件の関係が暗黙のうちに *OR* で評価されます。チェックされているときは、*AND* で評価されます。

「**追加**」ボタンを押して、単純条件を追加します。次に、左端のドロップダウンから「First」属性を選択し、さらに「has value(s)」演算子を選択します。この条件を否定するには、「**否定**」の列の値をトグルします。「has value(s)」演算子を使用するときは、右端のフィールドに何も指定する必要はありません。続いて、類似の、*has no values* 条件を「Last」という名前の属性にも追加します。最後に、「**すべてに一致**」のチェック・ボックスのチェックが解除されていることを確認してください。これはいずれかと一致 を意味します。これによって、どちらかの属性の値が欠落していると、このブランチがトリガーされます。

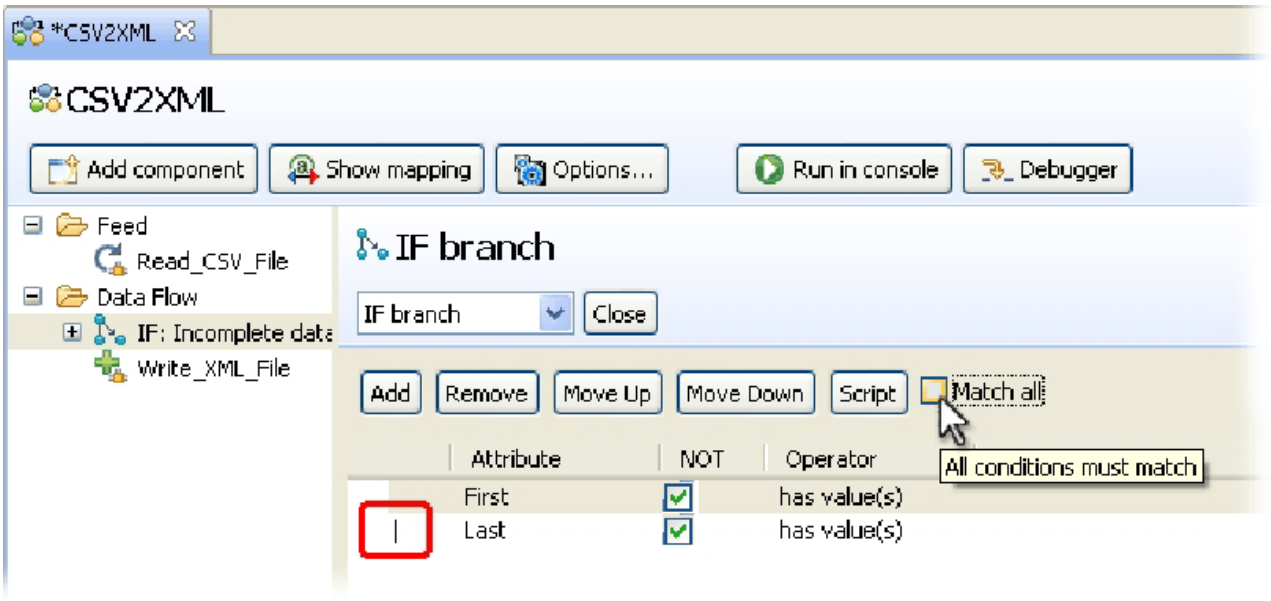

図 37. IF ブランチへの単純条件の追加

IF ブランチは、指定された条件の評価が *true* の場合、必ず AssemblyLine の実行フローの進路を変えます。今回は、「First」または「Last」属性のいずれかに値が割り当てられていないか、またはいずれも作業項目に存在しない場合には、ブランチの下に配置されたコンポーネントへの処理が続行されます。すなわち、「has value(s)」演算子は「exists」のチェックも行います。

ここで IF ブランチを展開して、その下に表示されるプレースホルダーをダブルクリックし、コンポーネントを挿入します。「**コンポーネントの選択**」ダイアログで、「**スクリプト**」のラジオ・ボタンをクリックし、「Script」という名前のコンポーネントを選択して「**終了**」を押します。このスクリプト・コンポーネント (「SC」とも呼ばれます) の名前を「Write to log」に変更して、以下の JavaScript のスニペットを中に入力します[23]。

```
task.logmsg("*** Skipping incomplete entry");
```

ここで使用する *task* 変数は AssemblyLine そのものを参照しており、 `logmsg()` メソッドのような有用な関数をいくつか提供します。このスクリプト化された呼び出しによって、指定したテキスト・メッセージがログ出力に書き込まれます。

ログ出力をより有用なものとするために、作業項目の現在の内容も出力されるようにします。これには、 IF ブランチを右クリックして、「**コンポーネントの追加...**」を選択し、「**スクリプト**」のラジオ・ボタンをもう一度選択します。今回は、「Dump Work Entry」というラベルが付いている定義済みのスクリプト・コンポーネントを選択します。

23. スクリプト・エディターには、選択可能な選択肢を表示する、*コード自動完成* という機能があります。例えば、スクリプト・エディターに「tas」と入力し、Ctrl + Space の組み合わせでキーを押すと、コード自動完成のドロップダウンが開き、「**task**」という選択肢が 1 つ表示されます。 Enter (キー) を押すと、この選択肢が選ばれてスクリプトに入力されます。ここでピリオド・キー (.) を押します。これで、スクリプトは「task」となります。しばらくして、今度は `task` オブジェクトでアクセス可能なすべてのメソッドやプロパティーを一覧にした、新規のコード自動完成のドロップダウンが現れます。

最後に、現在のサイクルをこの時点で中止して制御をイテレーター・コネクターに戻すよう AL に指示します。それによって、イテレーター・コネクターは CSV の次の行を読み取りに行くので、結果的に現在の項目が XML 出力からフィルタリングされます。この動作を自分で指定しない限り、制御は IF ブランチの後の最初のコンポーネントに移動します。そこで、IF ブランチをもう一度右クリックし、「**コンポーネントの追加...**」から「**スクリプト**」を選択し、次に「Exit Flow」という名前の SC を選択します。今度は、AL の表示がこのようになります。

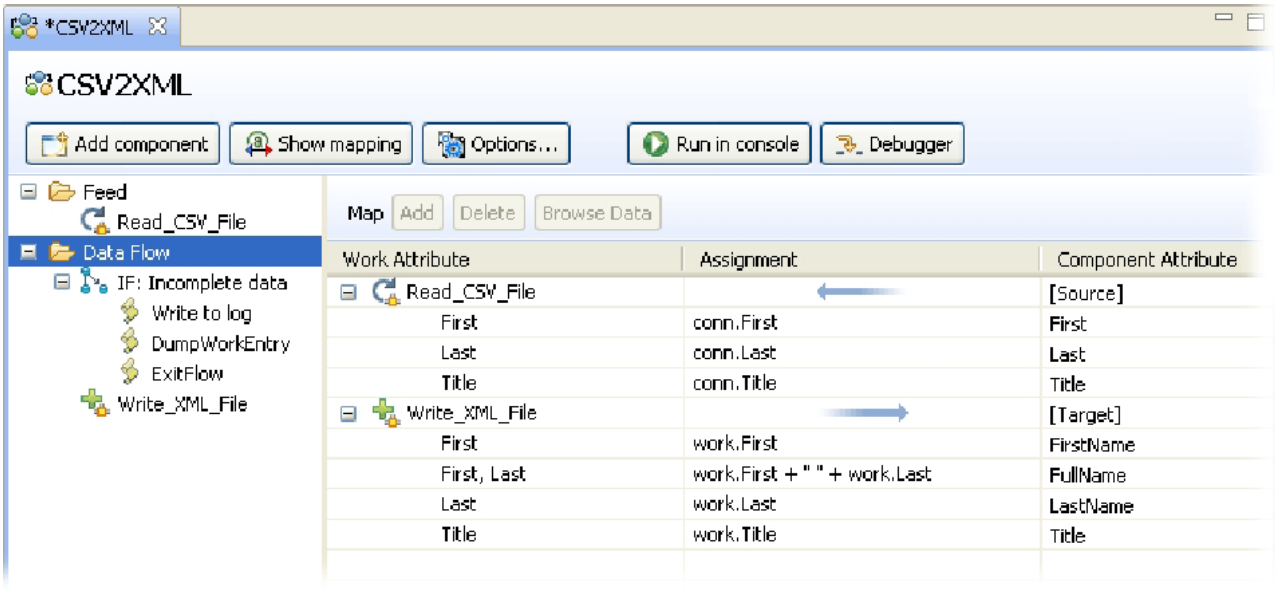

図 38. 最初の完全な AssemblyLine

AssemblyLine を再実行する前に、NULL 動作ダイアログをもう一度開いて、選択した定義と動作の両方をデフォルトに復元してください。そうしないと、IF ブランチの条件が *true* に評価されなくなります。

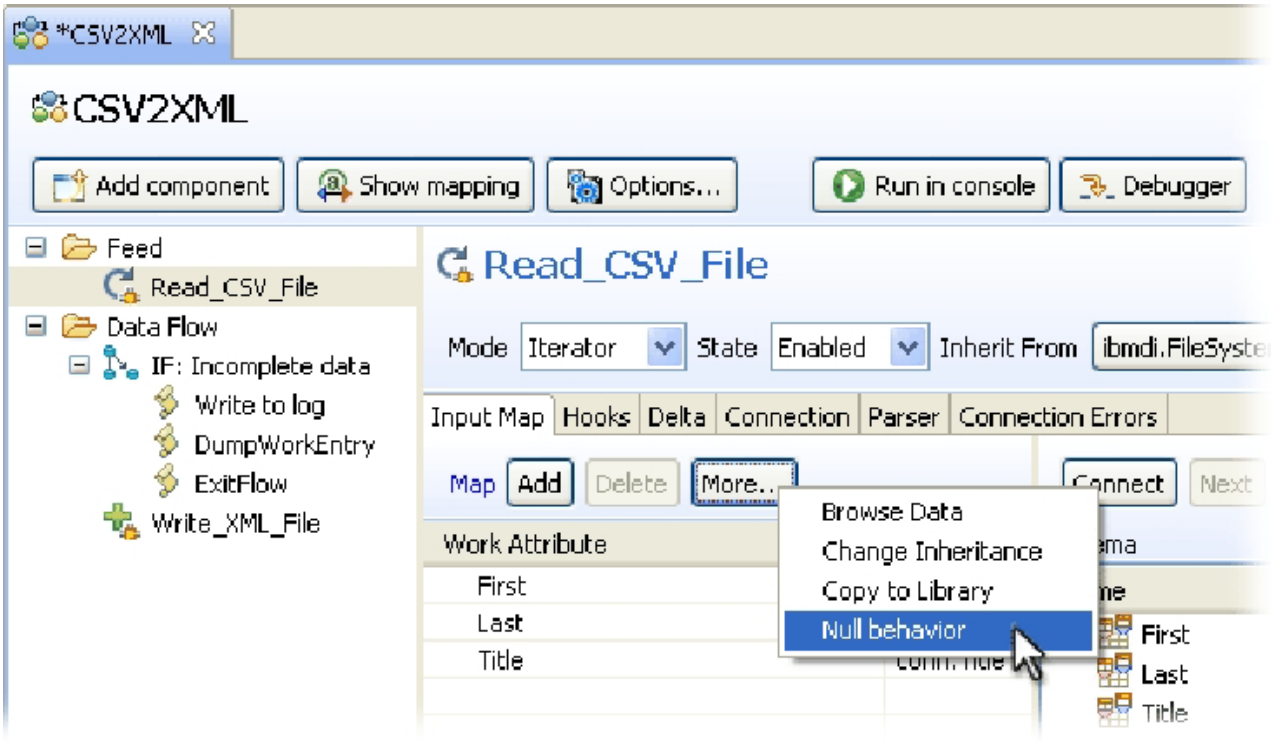

図 39. AssemblyLine の Null 動作のリセット

AssemblyLine を再実行すると、メッセージに続いて作業項目のダンプが表示されます。

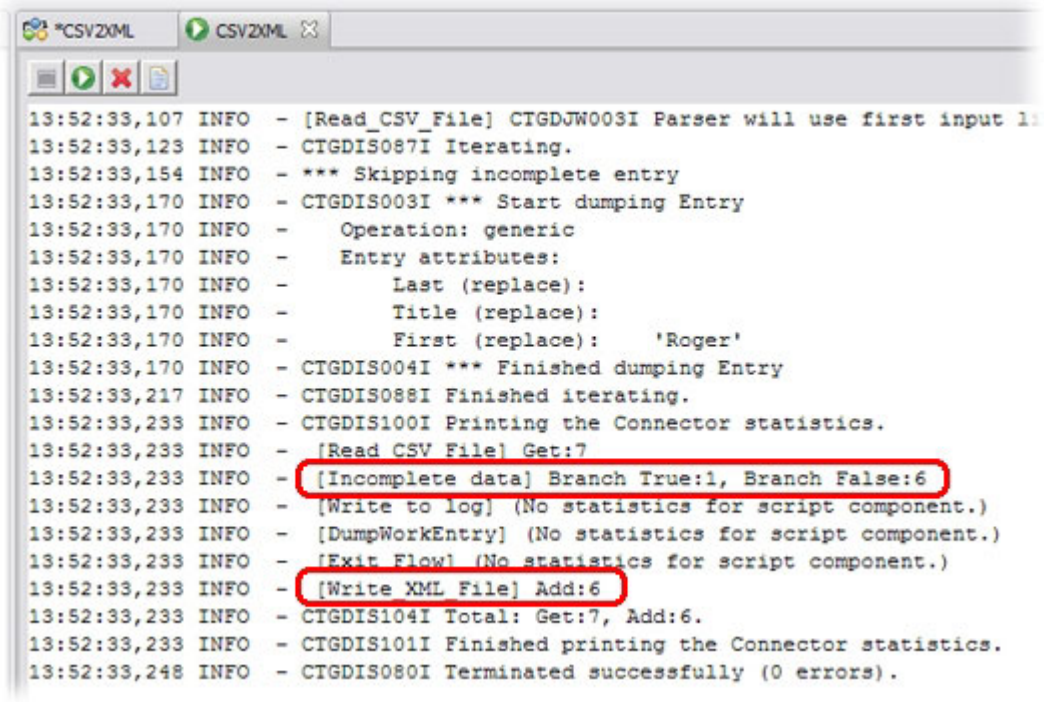

図 40. メッセージと作業項目のダンプを含むログ出力

この統計情報は、IF「Data incomplete」のブランチで true になったのが 1 回で、結果として 6 個のノードのみが XML 出力に追加されたことを示しています。データ・ブラウザー・ウィンドウをリフレッシュすると、「Roger」の表示が実際に消えています。

AssemblyLine のデバッグ

IBM Security Directory Integrator の最も強力な機能の一つは、AssemblyLine デバッガーが標準装備されていることです。このデバッガーによって、伝送途中のデータを表示したり、時には変更を加えたりしながら、AL の実行をウォークスルーすることができます。

では、「**デバッグ・セッションを開始します**」ボタンをクリックして、はじめての AssemblyLine をステップスルーしてみましょう。

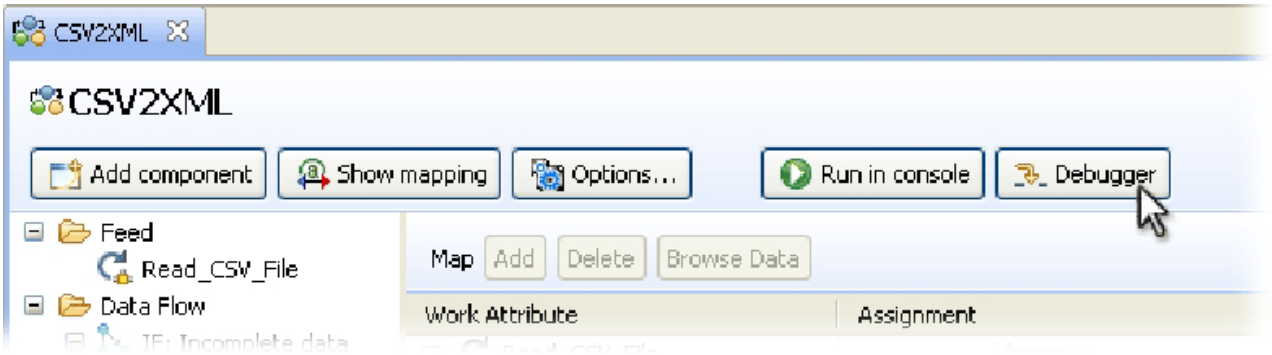

図 41. AssemblyLine のデバッグ

標準の「Run Console」ウィンドウの代わりに、Debugger のデバッグ用ウィンドウが表示されます。

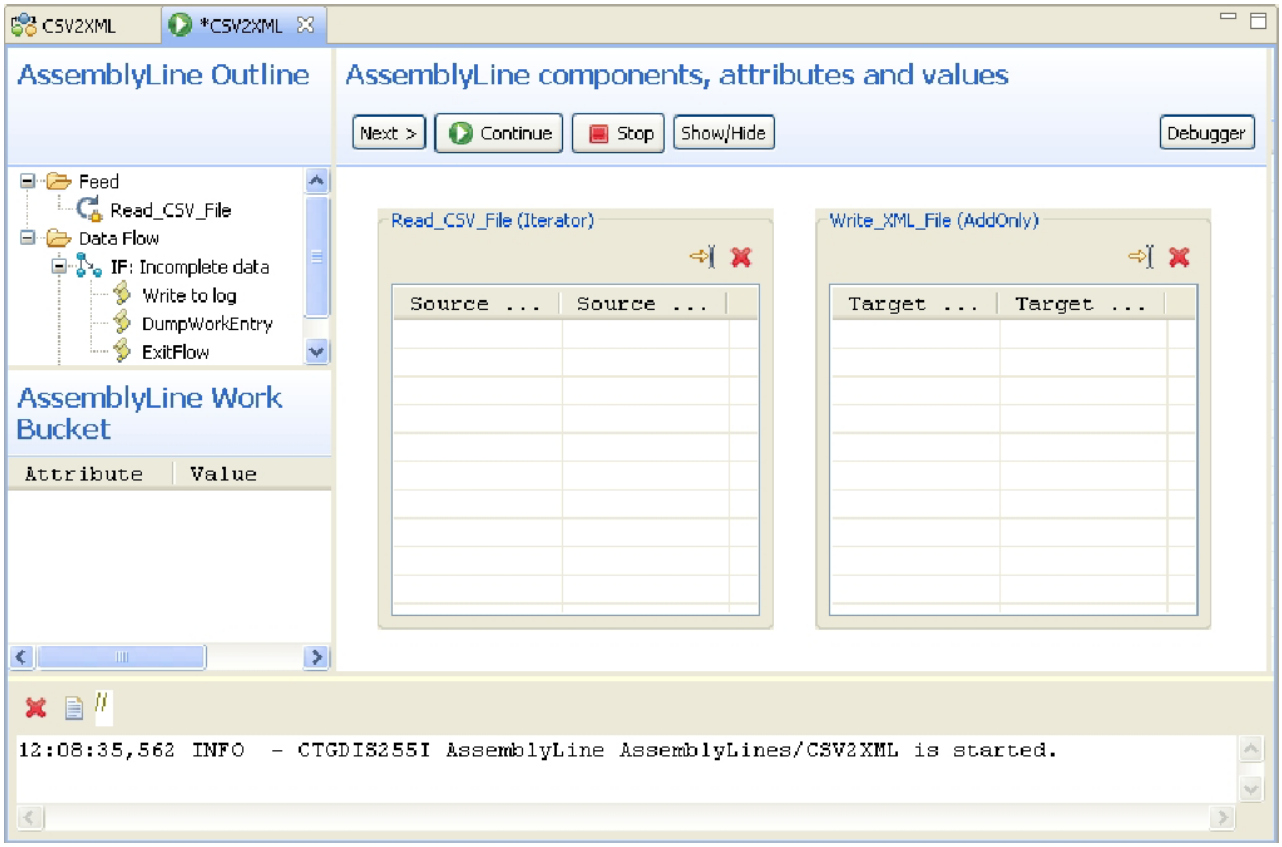

図 42. AssemblyLine のデータ・ステッパー

AssemblyLine のデバッガーには 2 つのモードがあります。シンプルで簡単なテスト機能を持つデータ・ステッパーと、スクリプトのステップスルー、Java ライブラリーの対話的操作、ファイル上のデータ変更のような、掘り下げたテストが可能な高機能の AL デバッガーです。

データ・ステッパーは、AssemblyLine コネクターの実行をステップ・スルーで行ったり、データの読み取り、書き込み、および変換の内容を表示するのに便利なツールです。この画面は、主に 3 つの領域に分割されています。

- 「**AssemblyLine の概要**」には AL が表示されます。実行を一時停止している箇所が強調表示されます。
- 「**AssemblyLine 作業バケット**」には、作業項目にマップされたすべての属性、つまり、入力マップ、または属性マップのコンポーネントにある属性が表示されます。
- 「**AssemblyLine コンポーネント、属性、および値**」には、デバッグ・セッションを制御するボタンが並んでおり、AL のすべてのコネクターのデータ表示グリッドがあります。

ウィンドウ下部には、コンソールの出力ウィンドウがあります。ここには、「実行」ボタンを使用して AssemblyLine を実行した場合に得られるのと同じ情報が表示されます。

この時点で、AssemblyLine はテスト・サーバーにディスパッチされており、コマンドでの実行を開始する準備ができています。 「次へ」ボタンを押すと、ステップ実

行を開始できます。

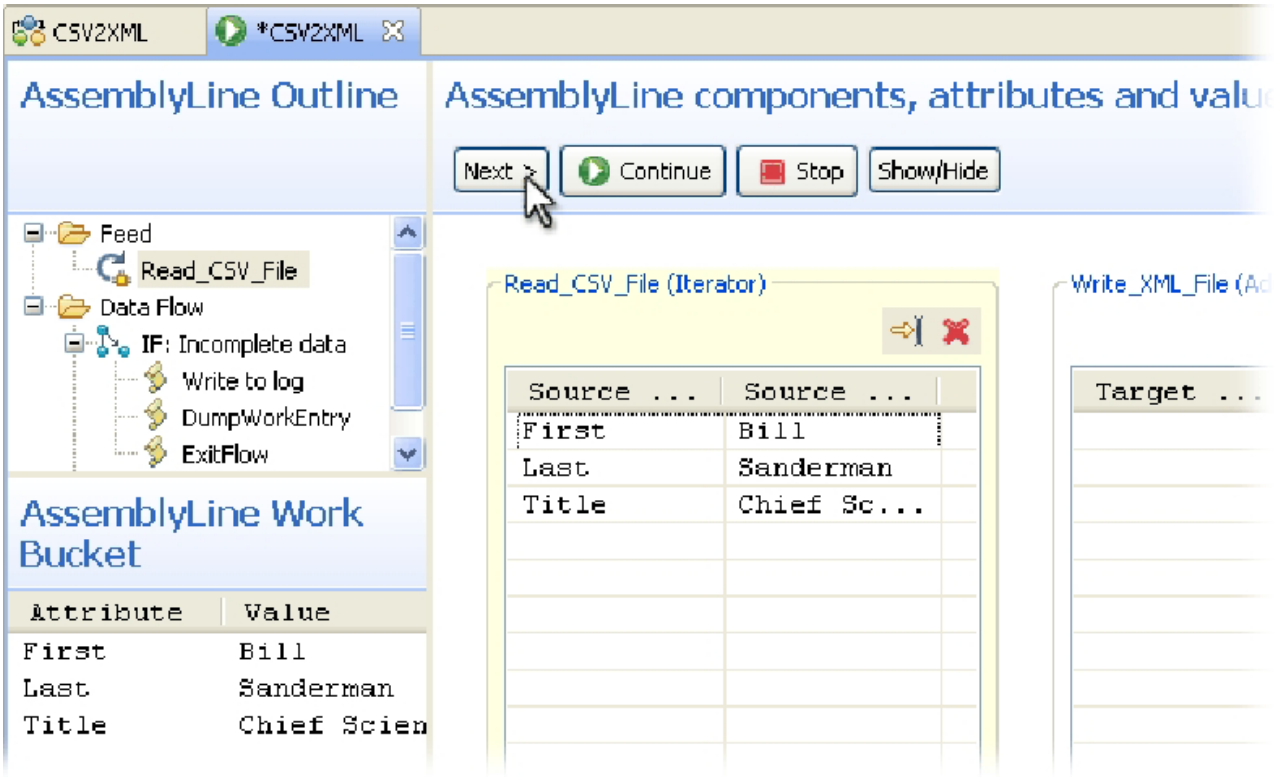

図 43. ステップイントゥ

スクリーン内で 3 つのことがどのように起こるのか、注意して見ていきます。「**AssemblyLine の概要**」では、「Read_CSV_File」コネクターが現在アクティブであることが示されています。「**AssemblyLine 作業バケット**」には、このコネクターに読み取られたばかりの属性が表示されます。また、これらの属性は、このコネクターのデータ表示グリッドにも取り込まれます。 「**次へ**」ボタンを押すたびに、次のステップが実行され、表示される情報がリフレッシュされます。

コネクターのデータ・グリッドの上部にある「**Run To Here**」ボタンを使用して、その場所までジャンプすることもできます。「Write_XML_File」コネクターに対してこれを実行します。

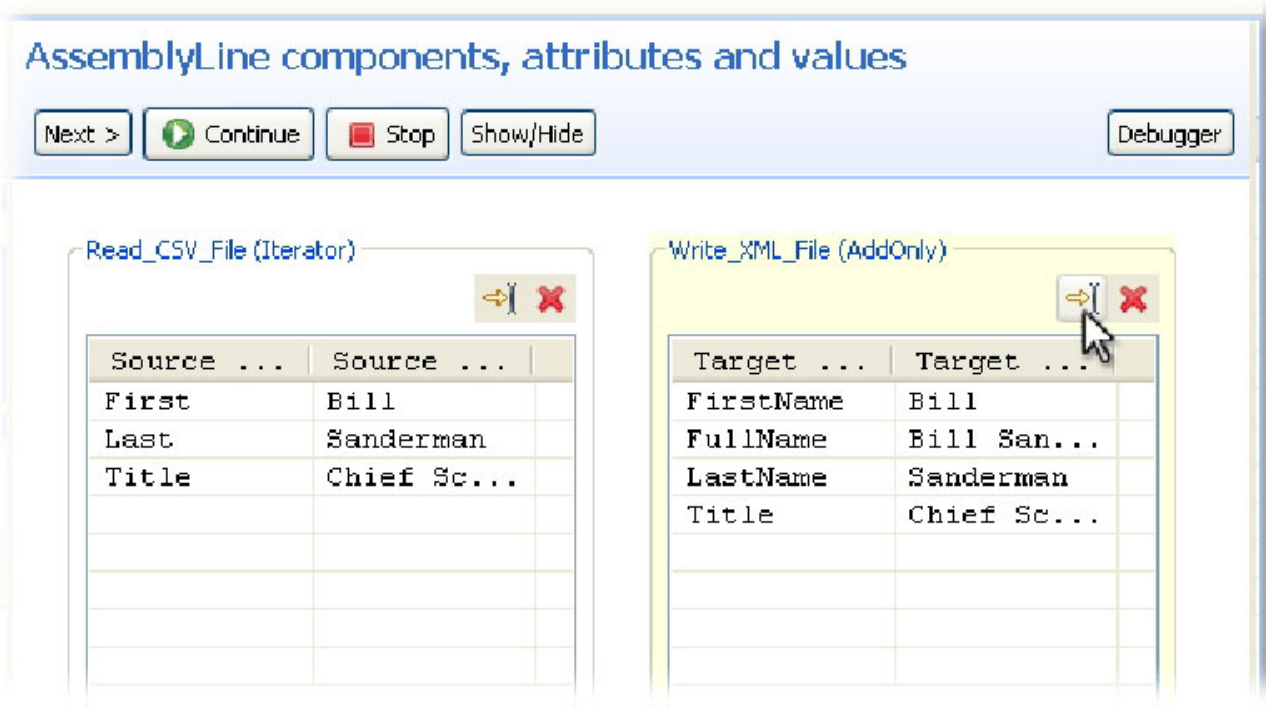

図 44. Write_XML_File コネクターまでステップ実行

この時、このデータ・グリッドには、このコネクターの出力マップの属性とその値が表示され、「FullName」の計算された値も表示されます。

それでは、データ・ステッパーのツールバー・ボタンを使用してどのような操作を行えるか見てみましょう。

- 「**次へ >**」は、次のコンポーネントの処理に移動し、すべてのデータ表示領域を更新します。
- 「**継続**」は、AL が終了するまで実行を継続します。
- 「**停止**」は、実行をすぐに終了します。
- 「**表示/非表示**」は、表示するコネクターのデータ・グリッドを決定できます。
- 「**デバッガー**」は、フル・デバッガー・モードに切り替えます。このモードでは、スクリプトをステップスルーしたり、ブレークポイントを設定したり、属性やスクリプト変数を表示および変更したり、実行中の AssemblyLine のコンテキストで JavaScript コマンドを対話式に実行することができます。

データ・ステッパーは、AssemblyLine がどのように動作するかについて、豊富な情報を提供しますが、拡張デバッガーの追加機能が必要な場合もあります。デバッグ・セッション時に必要に応じて、ステッパー・モードと拡張モードを切り替えることができることを覚えておいてください。では、データ・ステッパーのボタン群の右端にある「**デバッガー**」ボタンを押して、モードを切り替えてみます。

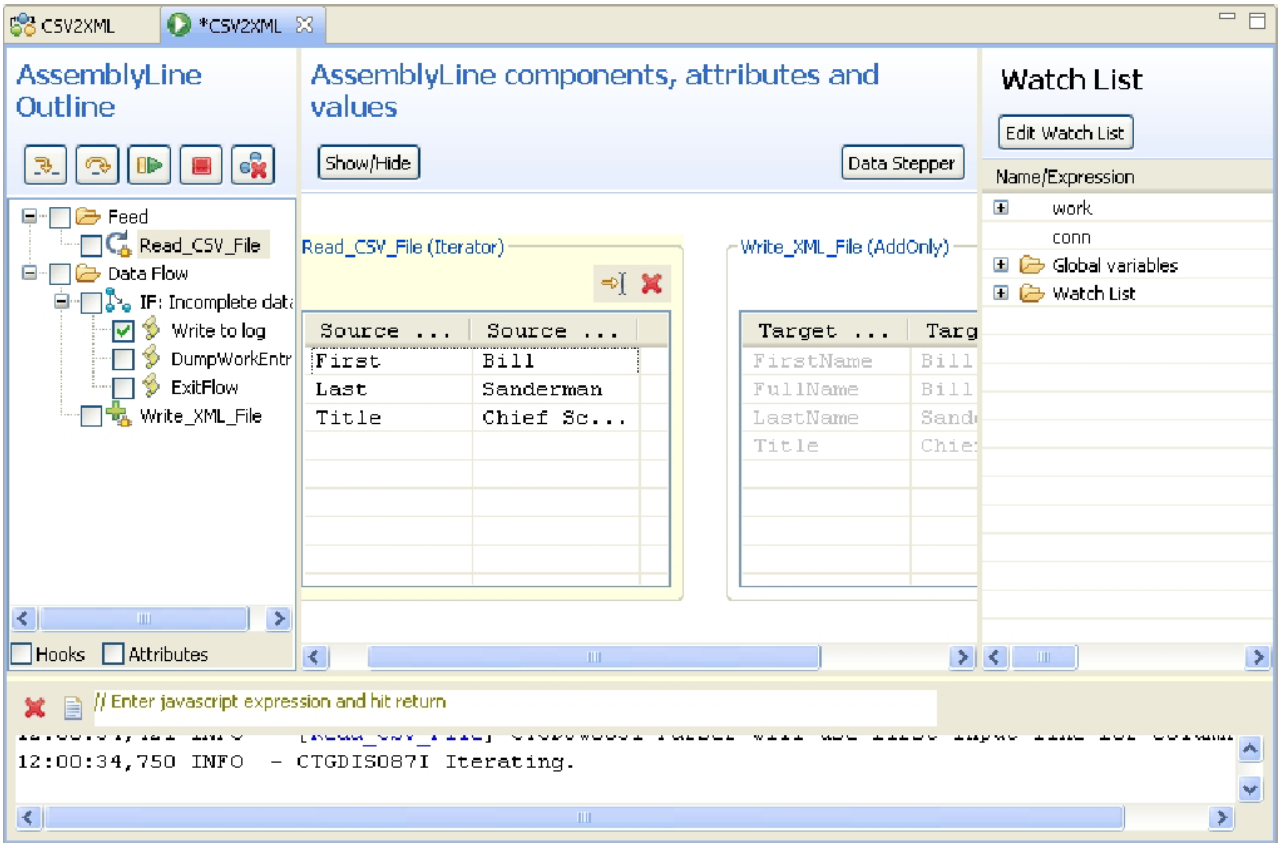

図 45. 拡張デバッガー・モード

モードを切り替えると、画面が再描画され、「**AssemblyLine の概要**」の新しいコントロールが表示されます。また、「**AssemblyLine 作業バケット**」が「**監視リスト**」に換わり、ウィンドウの右側に表示されます。「**監視リスト**」には、標準の属性「バケット」(work および conn) が表示されます。他に、「グローバル変数」と呼ばれるフォルダーも表示されます。これを開くと、 AssemblyLine 用に定義されたすべての変数、つまり「**work**」および「**system**」のような組み込み変数と、スクリプト・コードで定義した変数の両方が表示されます。最後にある「監視リスト」フォルダーは、自由に使用できます。監視したい変数や JavaScript の式全体を、このパネルの上部にある「**監視リストの編集**」ボタンを使用して追加することができます。

次に、「**AssemblyLine** の概要」を見てみましょう。このツリー表示内のコンポーネントの隣にあるボックスは、ブレークポイントと呼ばれ、これらの 1 つをクリックすると、IBM Security Directory Integrator を実行中に任意のコンポーネントで一時停止させることができます。また、任意のノード上で右クリックし、「**実行してここで中断する**」を選択して、AL の実行をそのポイントに移動することもできます。 概要の上のツールバーには、データ・ステッパーにあったものと同じものに加え、新しいコントロールもいくつか表示されます。

図 46. デバッガーのボタン

左端のボタンから説明します。

- 「**ステップイントゥ**」を使用すると、属性マップおよびスクリプトにステップインしたり、AssemblyLine およびそのコンポーネントの内部のワークフローにまでステップインしたりすることができます。これらの組み込みのフローの中間点を「フック」と呼びます。これについては、後の演習で説明します。
- 「**ステップオーバー**」は、データ・ステッパーにあった「**次へ >**」ボタンと同じです。デバッガーでは、これを使用して、スクリプト関数の中に入るのではなく、関数呼び出し後に停止させることができます。
- 「**継続**」は、AL が終了するまで (データ・ステッパーと同様)、または、ブレークポイントに達するまで実行します。
- 「**停止**」は、データ・ステッパーと同様で AssemblyLine を停止します。
- 「**Clear All Breakpoints**」は、AL に設定されているブレークポイントをすべて除去します。

ブレークポイントの働きを理解するために、「Write to log」スクリプトにブレークポイントを設定してみましょう。このコンポーネントの隣にあるボックスをクリックすると設定できます。

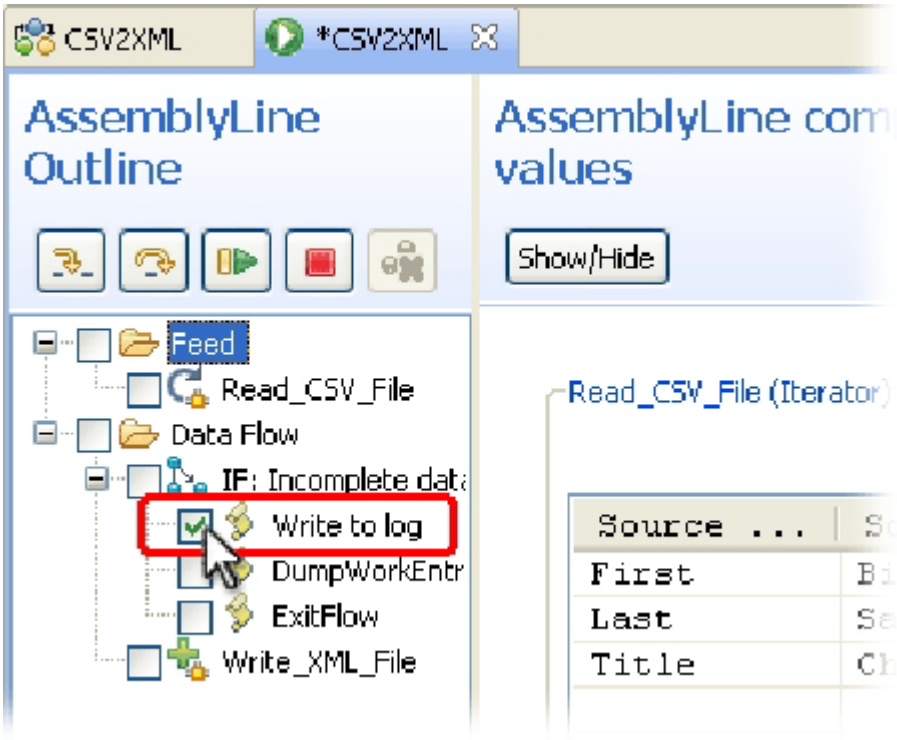

図 47. ブレークポイントの設定

この状態で「**継続**」ボタンを押すと、AL は IF ブランチが「真」になるまで実行され、「Write to log」スクリプトに制御が移動します。IBM Security Directory Integrator はスクリプト領域も開くので、ここでコードをステップスルーすることができます。ブレークポイントは、設定したい行の左側の余白をダブルクリックすることで、スクリプトの任意の行に設定できます。

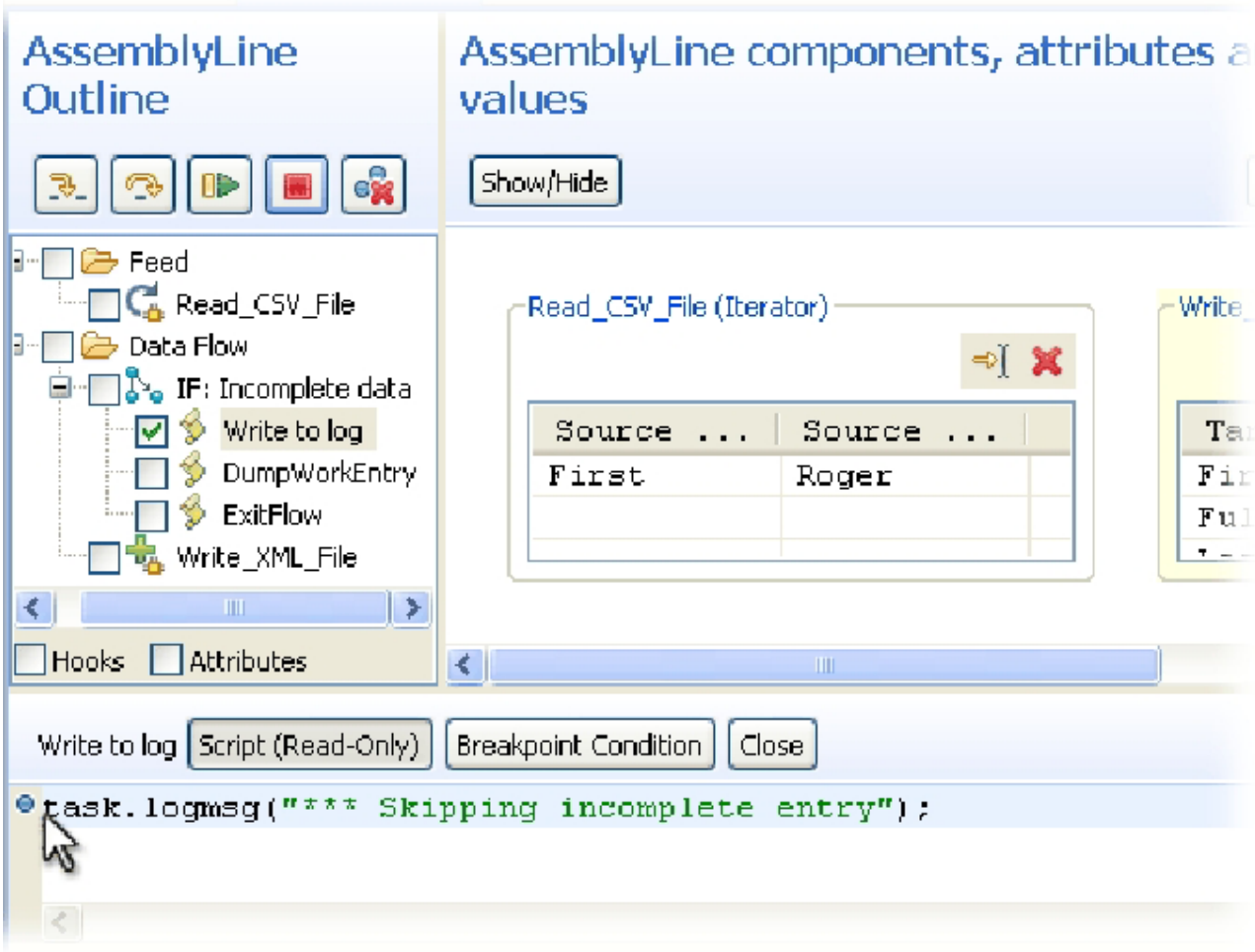

図 48. スクリプトでのブレークポイントの設定

さらに、コンポーネント・リストの任意のノードをダブルクリックすると、デバッグ領域を表示できます。上の図にあるように、「**ブレークポイント条件**」というボタンがあります。このボタンを使用して、*真* または *偽* かを評価する必要のある JavaScript の式を設定することができます。これにより、ブレークポイントがアクティブかどうかを決定できます。例えば、上に示されているブレークポイントは、次の場合に真になるように設定できます。

```
work.First.startsWith("R")
```

または

```
mycounter > 1000
```

これは、入力データ・セットの深いところでのみ発生する問題をデバッグする場合にとても便利です。

また、何らかの理由で、前のステップに戻る必要がある場合は、デバッグ・セッションを停止し、再始動するだけです。「データ・ステッパー」ボタンを押してデータ・ステッパーに戻ることもできます。

拡張デバッガーについての説明を終える前に、もう 1 つ注目すべき機能があります。**JavaScript 評価コマンド行**です。

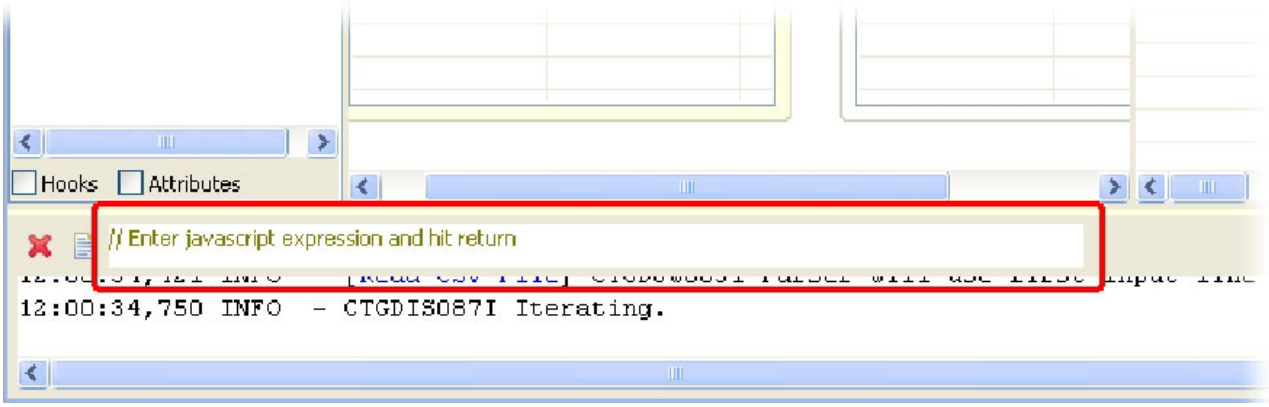

図 49. JavaScript 評価コマンド行

ログ出力領域上部の、この単純な入力フィールドでは、実行中の AssemblyLine のコンテキストに基づいて、スクリプトのスニペットを実行できます。次のコマンドを入力して Enter (キー) を押し、実行してみましょう。

```
task.dumpEntry(work)
```

これにより、作業項目の内容がログ出力ウィンドウに表示されます。次に以下のコマンドを実行してみます。

```
i = 42
```

ログに次のメッセージが表示されます。 `i=42 >> 42.0`

これは、新しい変数 (「i」) を定義してその値を 42 に設定したことを示します。式自体が (すべてのスクリプト・ステートメントと同様) 割り当てた値に評価されます。AL に既に定義されている変数および属性の値を変更することもできます。例えば、次のように入力します。

```
work.First="Rudy"
```

この行を実行した後、「First」属性の値は「Rudy」になります。伝送中のデータを変更できるということは、AssemblyLine がすべてのブランチ・ロジックまで動作することを確認し、ソリューションを完全にテストできるということを意味します。

AssemblyLine データ・ステッパーおよびデバッガーについて、時間をかけて十分に理解していただくことを強くお勧めします。それによって、IBM Security Directory Integrator サーバーのカーネルが提供するすべての組み込みワークフローを含めた AL の動作についてユニークな洞察が得られるだけでなく、その経験が独自のインプリメンテーションおよびデータに関するいくつかの仮定の検証に役立ちます。

順次ソースからのデータの検索

チュートリアルの練習の続きです。次のステップは、D2 からの検索の追加です。

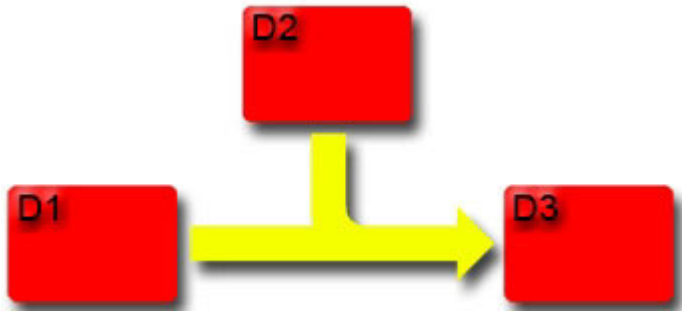

図 50. シナリオのフロー・ダイアグラム

チュートリアル・ディレクトリーには、D2 データ・ソースの役割を果たす `PhoneNumbers.xml` という名前のファイルが格納されています。このファイルには一連の XML 項目が保持されていて、それぞれの項目が属性「User」および「telephoneNo」を持ちます。

これから、出力の XML 文書に書き込むデータの一部として「telephoneNo」を含める作業を行います。このテキスト・ファイルに対しては、データベースやディレクトリーでは可能なランダム・アクセスによる検索ができないため、各 CSV 項目について正しい電話番号を検索するには、ファイルの中を繰り返し検索して、「User」と現在の CSV 項目の「FullName」を比較することになります。

しかし、「FullName」が最初に計算されるのは「Write_XML_File」コネクターの出力マップの中です。したがって、上の比較には間に合いません。つまり、この計算された属性を「Write_XML_File」の出力マップから「Read_CSV_File」の入力マップへ移動する必要があります。それにはまず、1 つのマップから別のマップへ属性マップ項目をドラッグします。

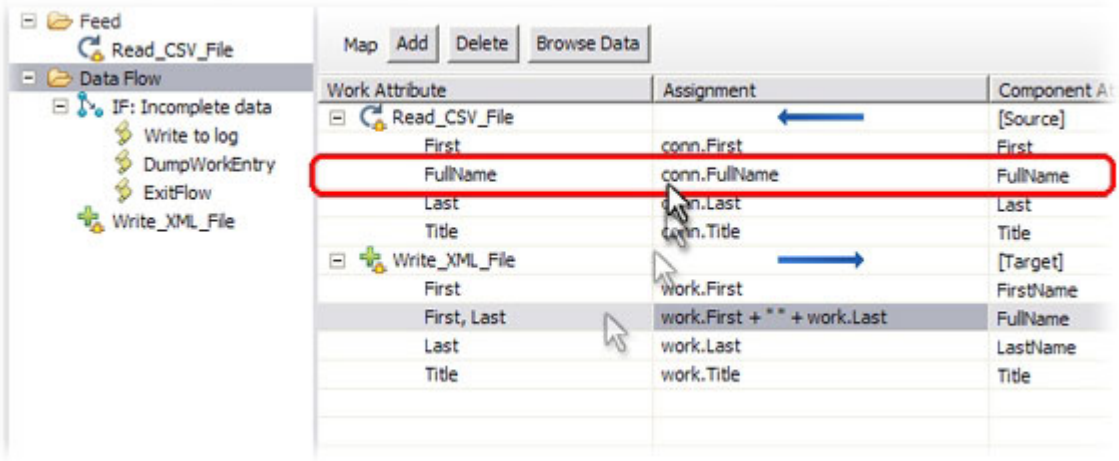

図 51. イテレーター・コネクターの入力マップへの「FullName」のドラッグ

これで、両方のマップに「FullName」マップ項目が入ります。ここで、入力マップの割り当てを調整する必要があります。 その理由は、ここでは、出力マップの場合のように作業項目から Conn 項目へマップするのではなく、Conn 項目から作業項目へマップするからです。この調整は、「Read_CSV_File」の下にある「FullName」をダブルクリックして、割り当てを次のように変更することで行います。

```
conn.First + " " + conn.Last
```

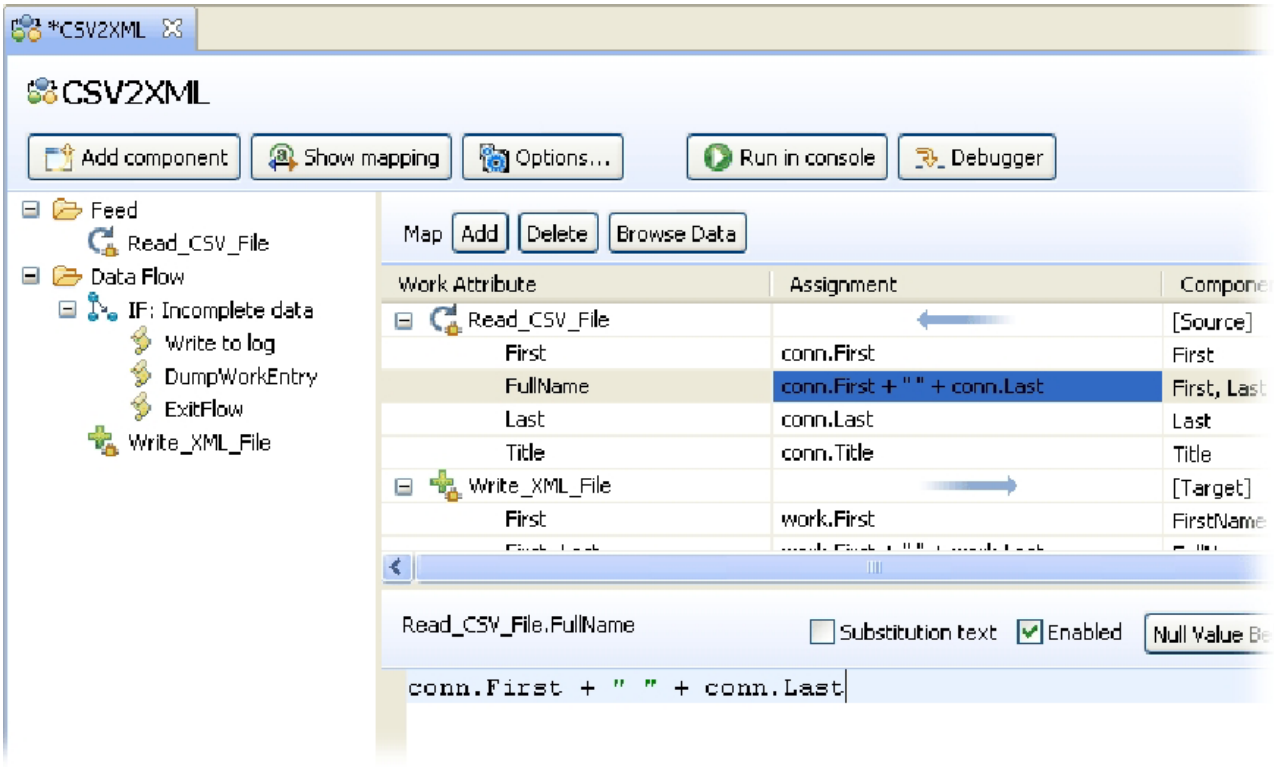

図 52. 「FullName」の割り当ての編集

元の出力マップの「FullName」項目を編集して、割り当てを単に `work.FullName` とすることもできます。これは、入力マップを変更したことにより、この属性が作業項目で使用可能になるためです。

ここで AssemblyLine を再実行して、`Output.xml` が変わっていないことを確認してください。確認が終わったら、ループ・コンポーネントを使用して `PhoneNumbers.xml` ファイルを読み取り、各ユーザーの番号を検索します[24]

まず、**「データ・フロー」**セクションに新規のコンポーネントを追加します。今回は *ConnectorLoop* という名前のコンポーネントを選んで、それに「TelephoneNumber」という名前を付けます。ConnectorLoop は、コネクターを使用してデータ・ソースから情報を読み取り、コネクターから返された項目ごとに 1 回ずつ、データ・ソースの下に接続されたすべてのコンポーネントに対して繰り返しアクセスするループ用のコンポーネントです。これは、フィード・セクションのイテレーター・コネクターの *for-each* 動作と似ています。この動作では、読み取られた項目ごとに、**「データ・フロー」**セクションのコンポーネントに繰り返しアクセスします。

24. ループ・コンポーネントには、次の 3 つのタイプがあります。1) *ConnectorLoop*。これは、イテレーターまたはルックアップ・モードのコネクターから返されたデータに対する繰り返しを可能にします。この練習では、このタイプのループを使用します。2) *ForEachAttributeValueLoop*。IBM Notes や LDAP Directory などのシステムに見られるような、複数の値を持つ属性のループ検索に便利です。3) *ConditionalLoop*。これは、ブランチで使用するのと同様な単純条件およびスクリプト記述条件を使用して、繰り返しを制御するものです。

作成した「TelephoneNumber」コネクター・ループを IF ブランチと「Write_XML_File」コネクターの間にドラッグします。これが、IF ブランチの*内側*に入らないように注意してください。

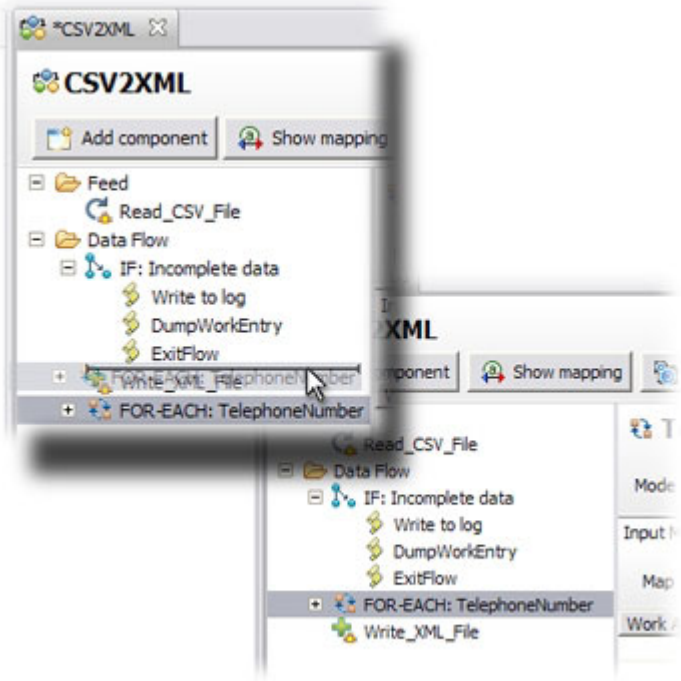

図 53. コネクター・ループのドラッグ

それを選択してエディターを開きます。このエディターは、コネクター・エディターに似ています。

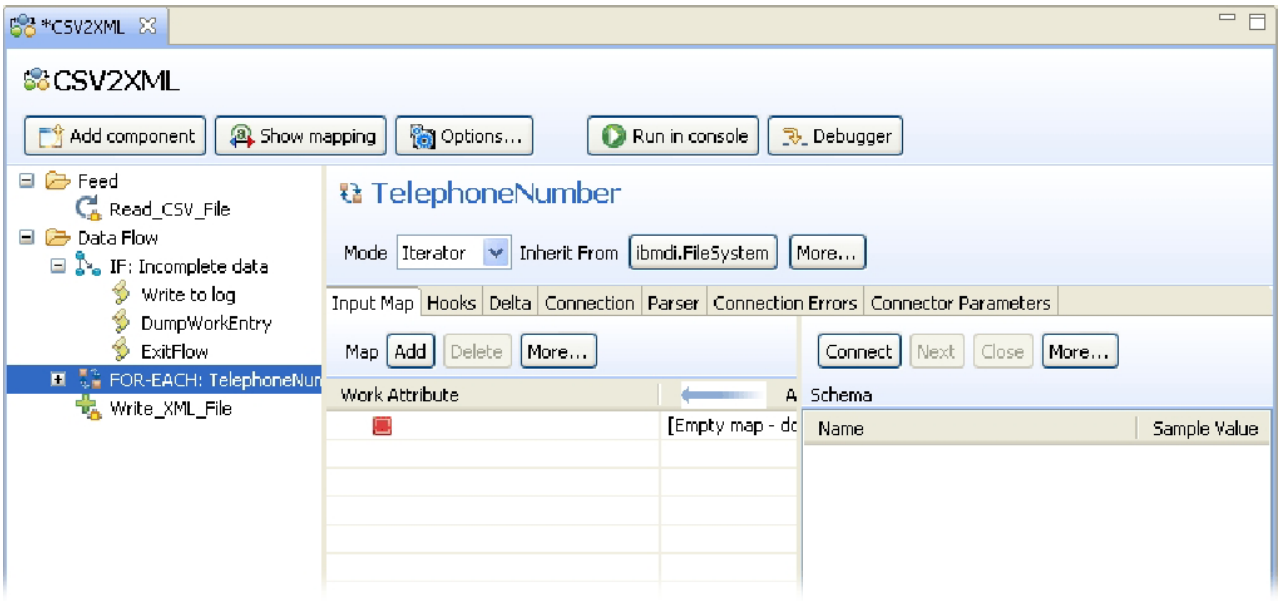

図 54. コネクター・ループの構成

主な違いは、「**モード**」ドロップダウンには「**イテレーター**」と「**ルックアップ**」のオプションしかないことです。そのほかに、「**その他...**」ボタンがあります。このボタンは、繰り返しの対象になる項目を制限するためのオプションと、3 つの選択肢のある「**初期化**」ドロップダウン・パラメーターを提供します。

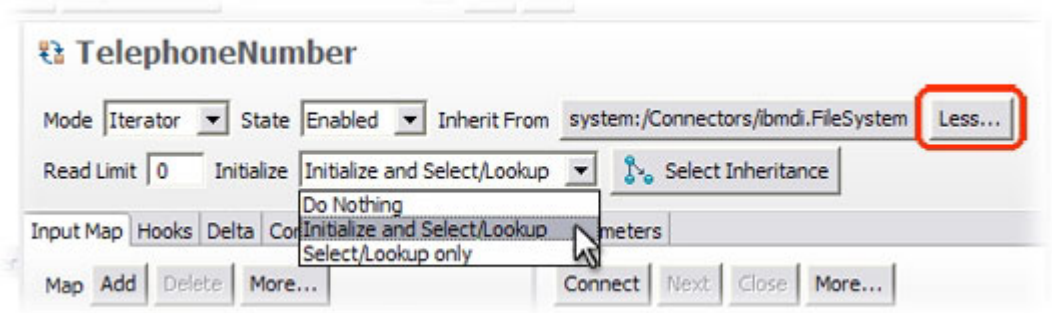

図 55. コネクター・ループの拡張設定

- 「**何も実行しない**」を選択すると、AL の処理中にこのループに到達しても、組み込みのコネクターは初期化されません。
- 「**初期化および選択/ルックアップ**」を選択すると、ループの繰り返しが開始されたときに必ずコネクターが初期化されます。使用するコネクター・ループはファイルを読み取り、毎回ファイルの先頭から開始する必要があるため、このオプションを使用します。
- 「**選択/ルックアップのみ**」は、コネクター・ループがデータベース、ディレクトリー、またはその他のランダム・アクセス可能なデータ・ソースを指している場合に有効です。この場合、接続を毎回初期化しなおす必要はありません。必要なのは、検索を再発行することだけです。検索とは、イテレーター・モードの選択操作、またはルックアップ・モードのルックアップ操作です。

PhoneNumber.xml ファイルを読み取るために、ループ・コネクター (デフォルトではタイプが「FileSystem」) を構成してから、「XML パーサー」を選択します。次に、属性マップのタブを開いて、属性をディスカバーします。

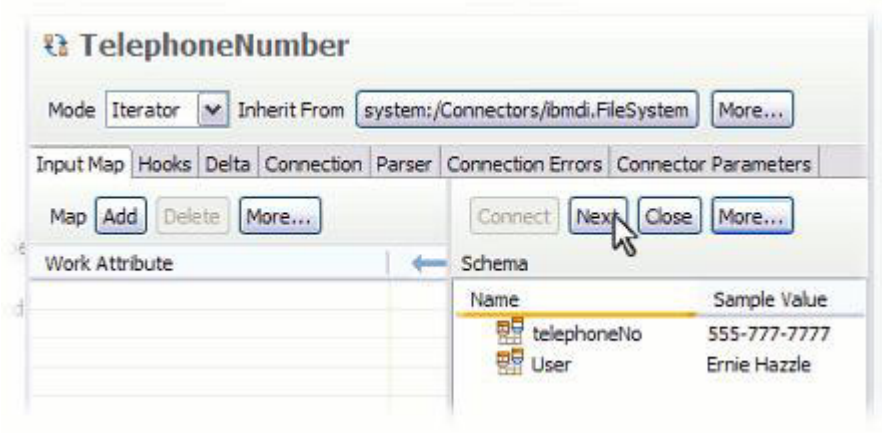

図 56. 階層的な属性

マップの作成は、属性レベルで、入力スキーマの中の「User」および「telephoneNo」を選択して入力マップへとドラッグすることにより行います。

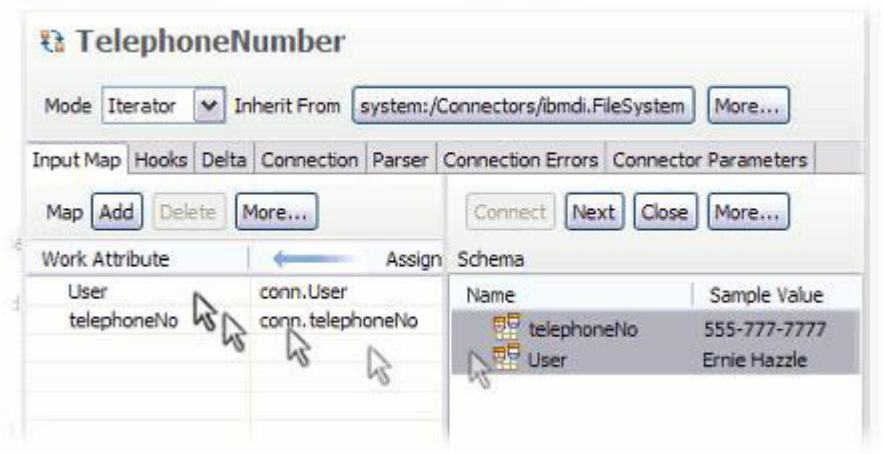

図 57. スキーマから属性マップへのドラッグ

操作の結果、属性「User」および「telephoneNo」に、1 つずつマップのルールが作成されます。

ここでループ・コネクター・エディターをクローズして、その下に IF ブランチを追加します。それを行うには、コネクター・ループを右クリックして、**コンポーネントの追加...**」を選択します。この IF ブランチの名前は、「Matching name found」とします。ここで、「User」と「$FullName」が *equals* であるかどうかを検査する単純条件を追加します[25]。

25. このドル記号は特殊文字で、この場合、「FullName」が照合に使用するリテラル・ストリングではなく、作業項目内にある属性の値であることを示しています。

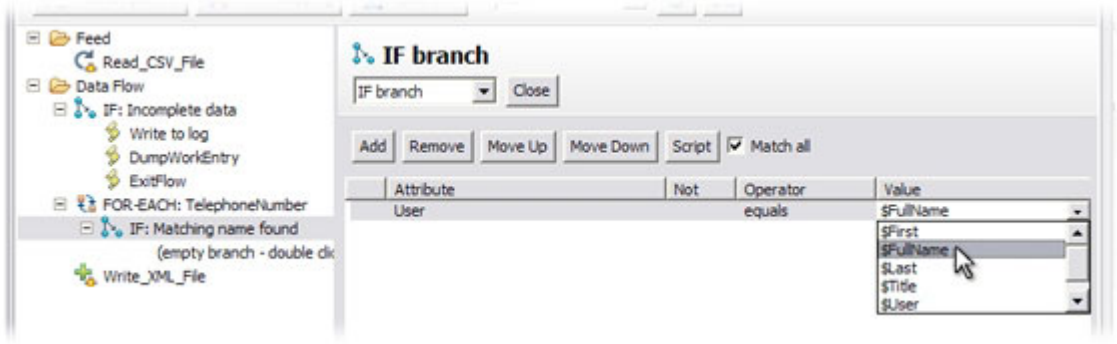

図 58. IF ブランチ用の条件エディター

一致が生じたときには、「User」および「telephoneNo」属性に正しい値が格納された状態でコネクター・ループを終了する必要があります。そのためには、「Exit loop」という名前をつけたスクリプト・コンポーネントを追加して、その中に次のスクリプトを記述します。

```
system.exitBranch("loop");
```

しかし、コネクター・ループが一致を検出しないまま `PhoneNumbers.xml` の終わりに到達した場合は、どうなるのでしょうか。「User」および「telephoneNo」属性にはファイルの最後の項目から読み取られた値が格納されているので、値が空の属性を検査してみてもこの状態は検出できません。照合の失敗を検出するための何らかの手段が必要になります。

解決策は、一致が検索されたことを示すフラグとして、スクリプト変数を使用することです。そのためにここで、「Found user」と名付けたスクリプト・コンポーネントを IF ブランチの中に挿入し、「Exit loop」スクリプト・コンポーネントの直前にドラッグします。このスクリプト・コンポーネントには次のスニペットが含まれます。

```
foundUser = true;
```

一致を検出しないまま入力ファイルの終わりに到達したことを示すには、コネクター・ループを選択して「フック」タブを開きます。

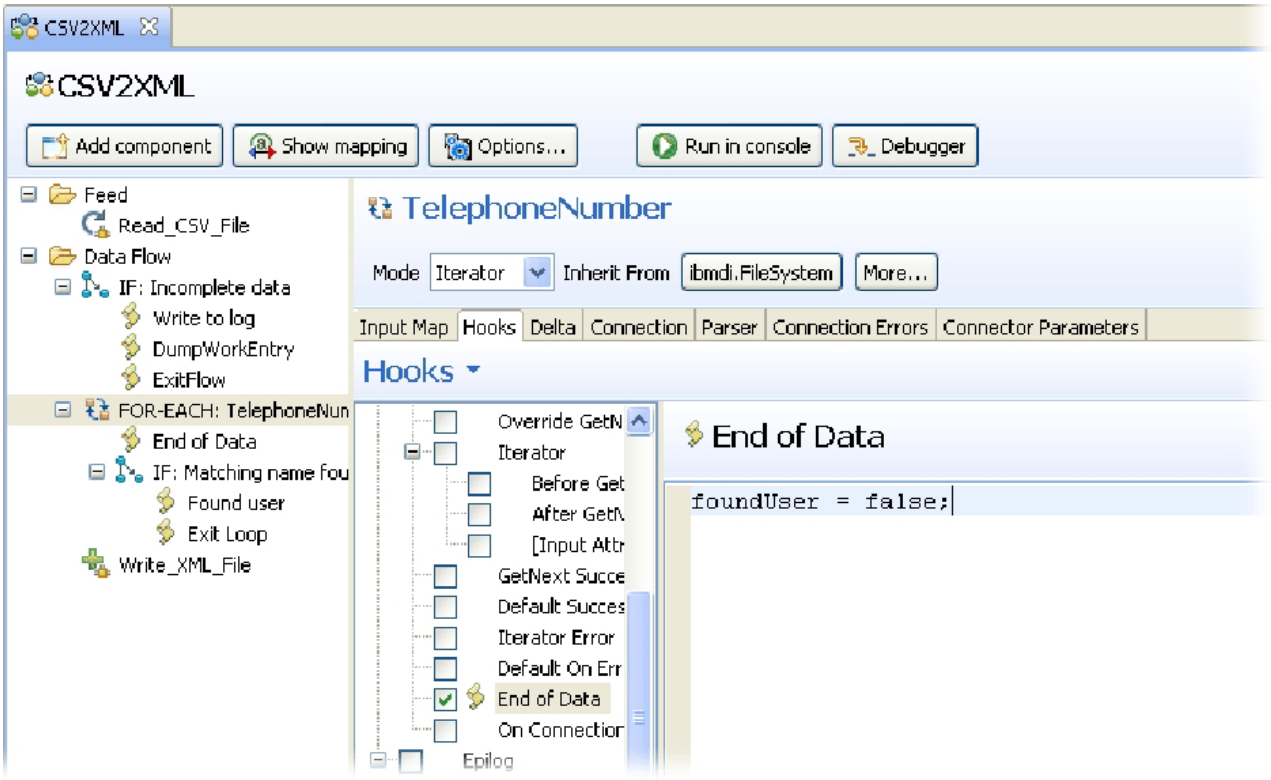

図 59. 「データの終わり」フックのスクリプトの記述

「データの終わり」という名前のフックを選択して、このスクリプトを入力します。

```
foundUser = false;
```

「データの終わり」フックに到達するのは、コネクターが、接続されたソースの最後の項目を越えて読み取りを試みた場合だけです。この場合、一致は検索されません。

これで、AssemblyLine のコンポーネントは次の図のようになります。

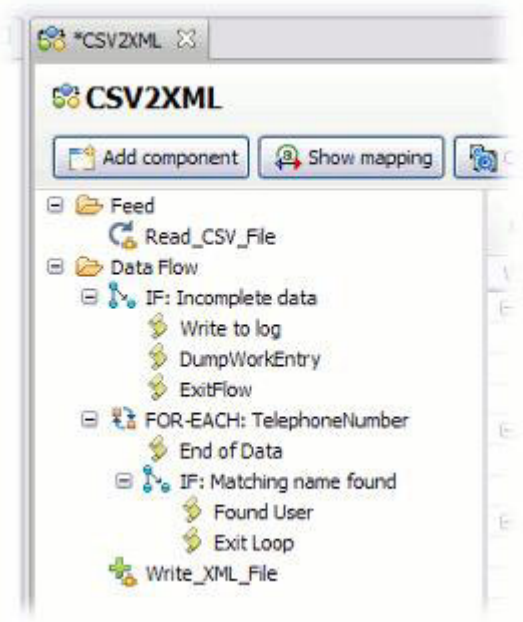

図 60. AssemblyLine のデータ・フロー部分のコンポーネント・リスト

これで、スクリプト変数を検査することにより、検索に成功したかどうかを確認できるようになりました。一致が検索されなかったときには「telephoneNo」属性にデフォルト値を設定する必要もあるため、このことは重要です。さもないと、コネクター・ループによって読み取られた最後の値が残されたままになります。

そこで、もう 1 つの IF ブランチをコネクター・ループの下に追加して、「NOT foundUser」と呼ぶことにします。「**IF ブランチ**」詳細領域の「**スクリプト**」ボタンをクリックして、スクリプト変数の値を検査する次のスクリプトを入力します。

```
! foundUser
```

感嘆符は foundUser の値を否定します。したがってこの値がコネクター・ループの「データの終わり」フックで *false* に設定されている場合には、このブランチ条件は *true* と評価されます。

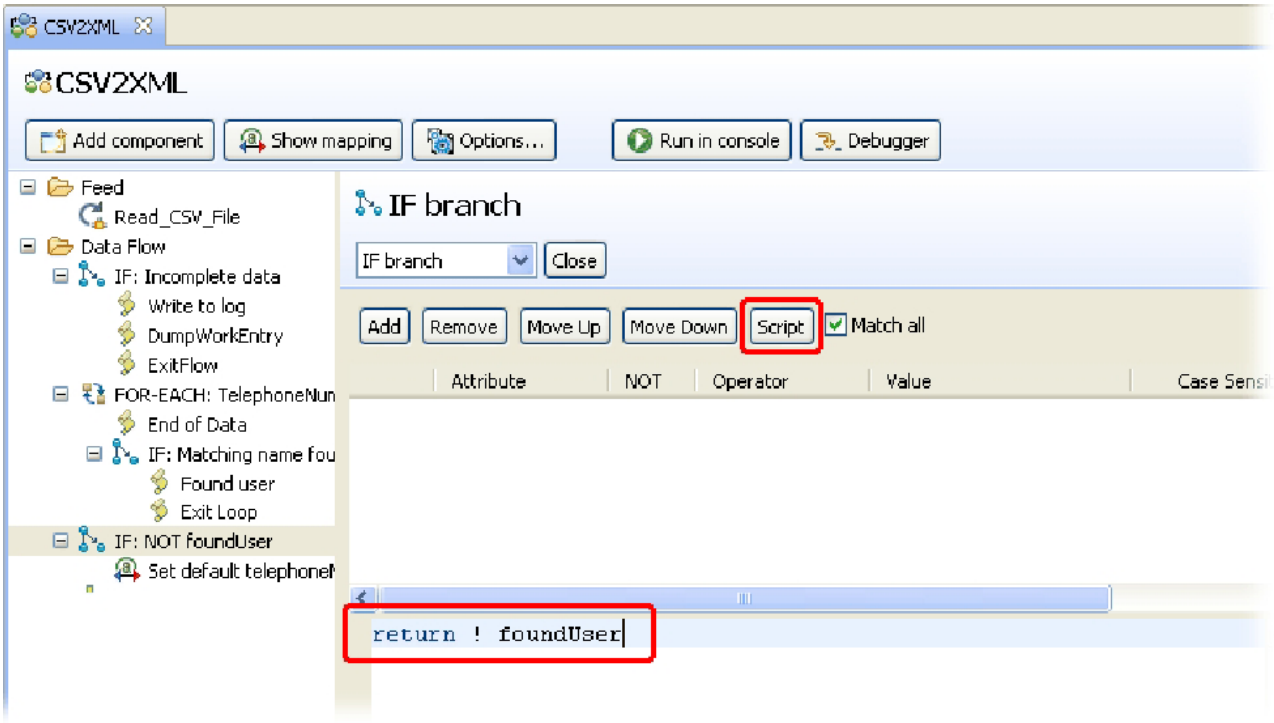

図 61. IF ブランチ条件のスクリプト記述

「Attribute Map」タイプの新規コンポーネントを「NOT FoundUSER」の直下に挿入します。この AssemblyLine のコンテキストにおける機能がわかるように、この属性マップ・コンポーネントを「Set default telephoneNo」と名付けます。ここで「**属性の追加**」ボタンを使用して、「telephoneNo」という名前の単一の属性を作成します。この名前は、コネクター・ループが返して来るものと同じです。割り当てスクリプトを設定するために、この属性をダブルクリックします。

```
「N/A」
```

これは、 CSV 入力から読み取られ、PhoneNumbers.xml で見つからなかった人物の「telephoneNo」の値が「N/A」になるという意味です[26]。

最後に、この新規の「telephoneNo」属性を「Write_XML_File」の出力マップにドラッグして、そこに加え、割り当てが次のようになっていることを確認します。

```
work.telephoneNo
```

AssemblyLine は次の図のようになるはずです。

26. そのようなユーザーに「telephoneNo」属性を与えないようにするには、値を返さない割り当てを使用します。そのためには、次のように特殊値 NULL を返します。

    ```
    NULL
    ```

    これによって、デフォルトの NULL 動作が作業項目から属性を取り除きます。その結果、属性は「Write_XML_File」コネクターの出力マップに到達せず、したがって出力の XML 文書にも現れません。

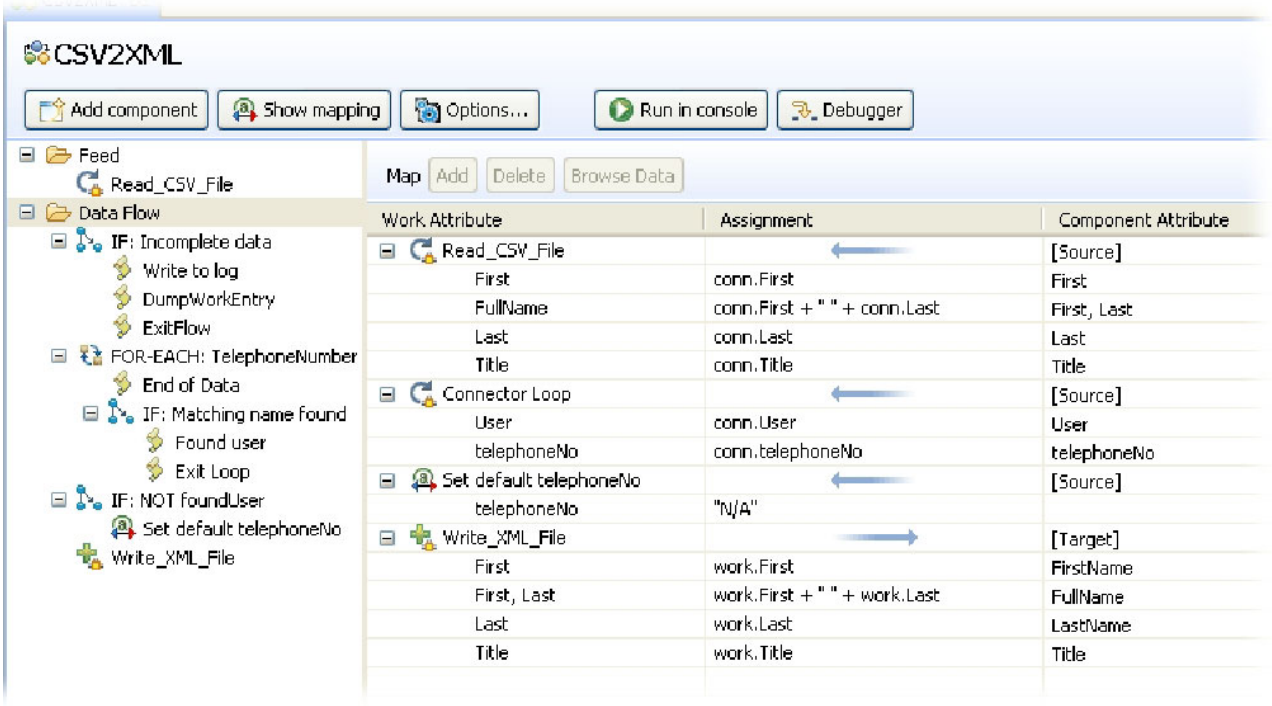

図 62. For-Each ループ付きの AssemblyLine

では、AL をもう一度実行して、ログ出力を点検してみてください。「NOT foundUser」ブランチが 2 回 *true* になっていて、他の 4 つの項目は *false* になっているはずです。

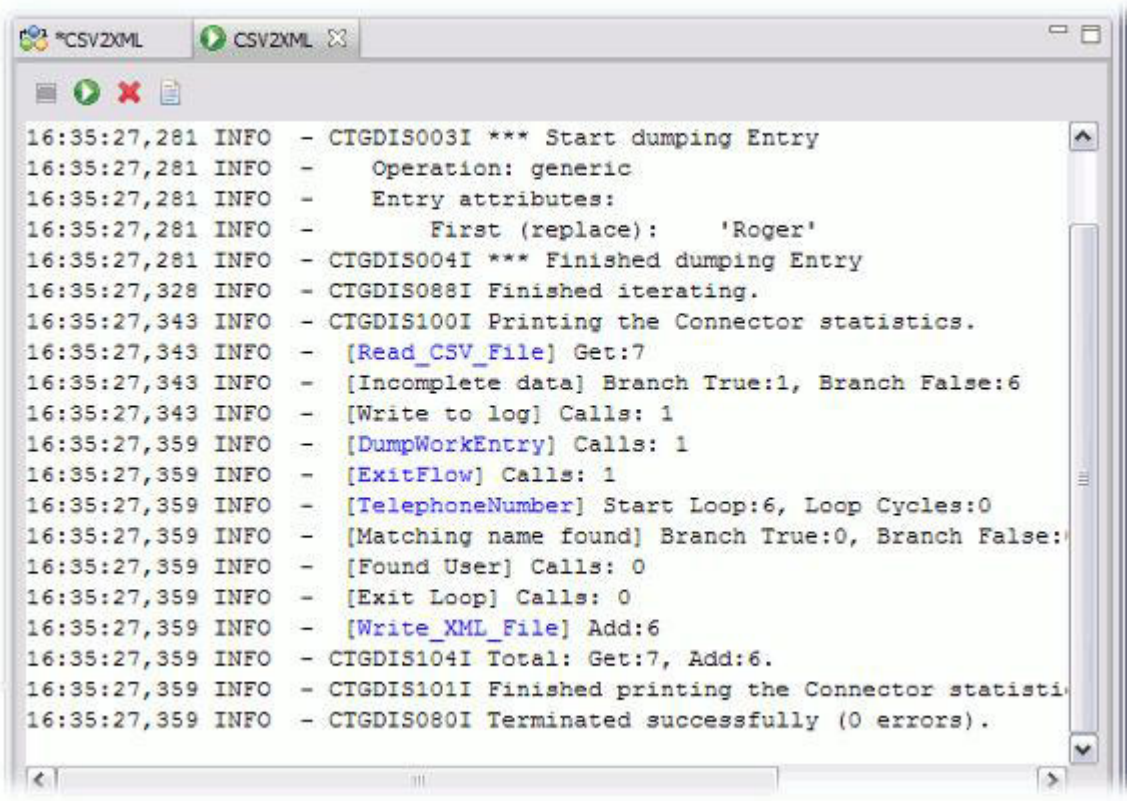

図 63. IF ブランチ統計付きのログ出力

このログ出力の AL 統計の中で、名前が強調表示 (青色表示) されているコンポーネントがあることに注意してください。そのいずれかの上で、Ctrl キーを押しながらマウスの左ボタンをクリックすると、選択されたコンポーネントが AssemblyLine エディターの中で開きます。

結果的に、XML 出力は次のようになります。

図 64. 「telephoneNo」属性が含まれる XML 出力

ここまではうまくいきました。引き続き、結合操作を行うためにルックアップ・モードを使用します。

ルックアップ・モードの使用

実行中の AssemblyLine を変更して混乱を招くという危険を冒す代わりに、AssemblyLine をコピーして変更します。これを行うには、「CSV2XML.assemblyline」を右クリックして「**コピー**」を選択します。

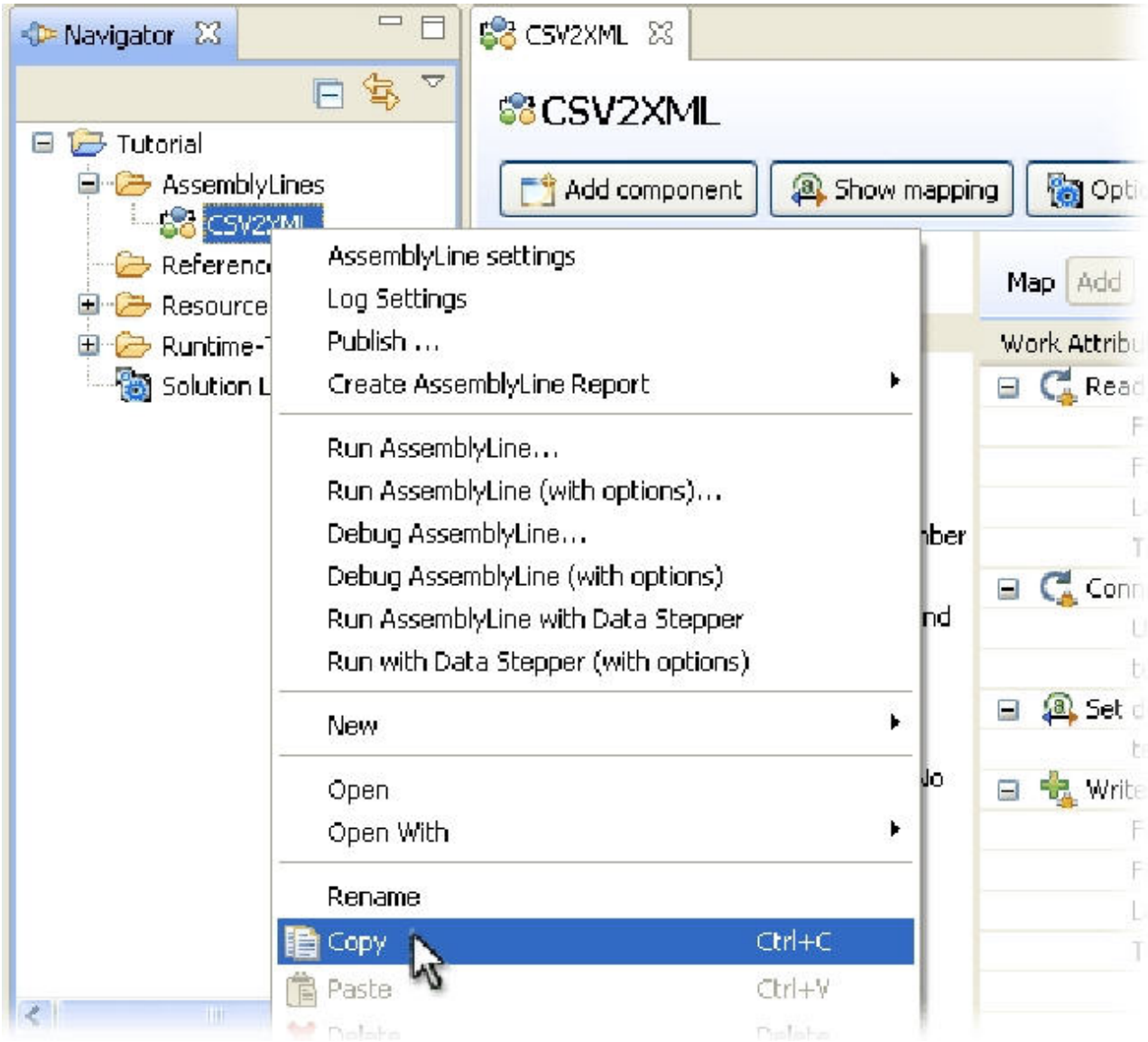

図 65. AssemblyLine コピー機能

ナビゲーターの「AssemblyLines」フォルダーを右クリックして、**「貼り付け」**オプションを選択してください。新しくできた AssemblyLine 名前を「CSV2XML_LookupMode」にし、ダブルクリックして AL エディターをオープンします。

ここでコネクター・ループをその下のすべてのコンポーネントと共に削除することができます。コネクター・ループを選択して **Delete** キーを押してください。この場所に、他の AssemblyLine からコピーした JDBC コネクターをドラッグします。

まず、このコネクターが読み取るデータベース表を作成する必要がありますが、データベース表を作成するためには、作成済みの AssemblyLine を実行する必要があります。これを行うには、ファイル・ブラウザーを使用して「Tutorials」フォルダーへとナビゲートし、`CreatePhoneDB.assemblyline` というファイルを見つけます。それをナビゲーター・パネルの「AssemblyLines」フォルダーの上の CE ウィンドウに

ドラッグします。

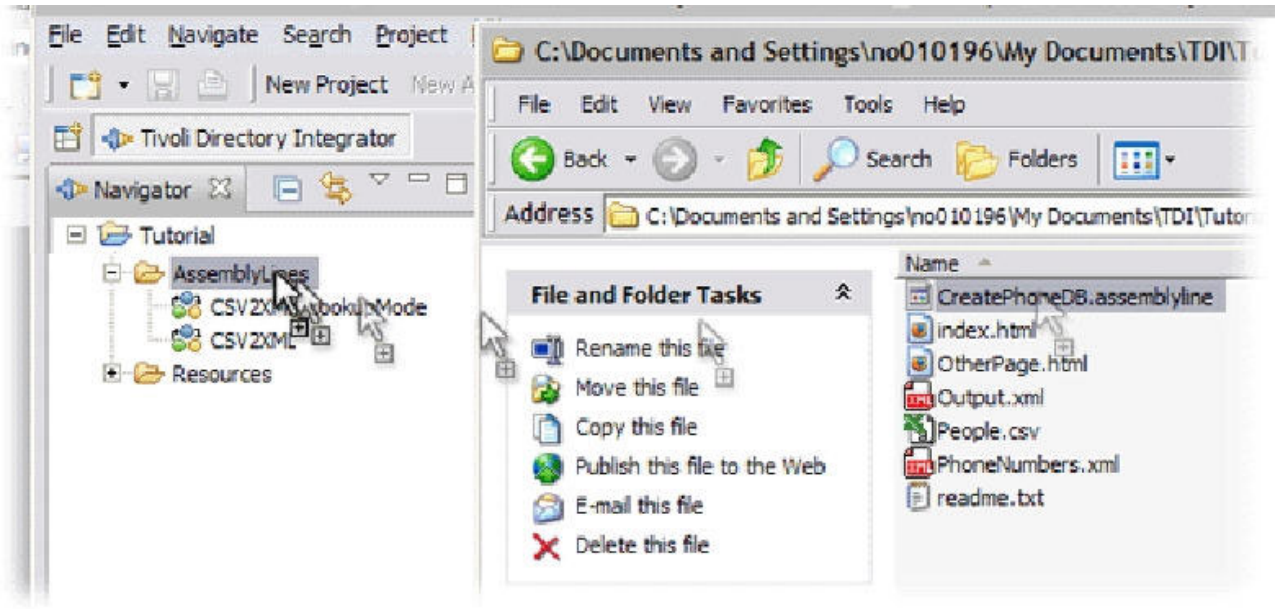

図 66. プロジェクトへの AssemblyLine のコピー

AssemblyLine がプロジェクトにインポートされます。これを右クリックして、「**AssemblyLine の実行...**」を選択します。

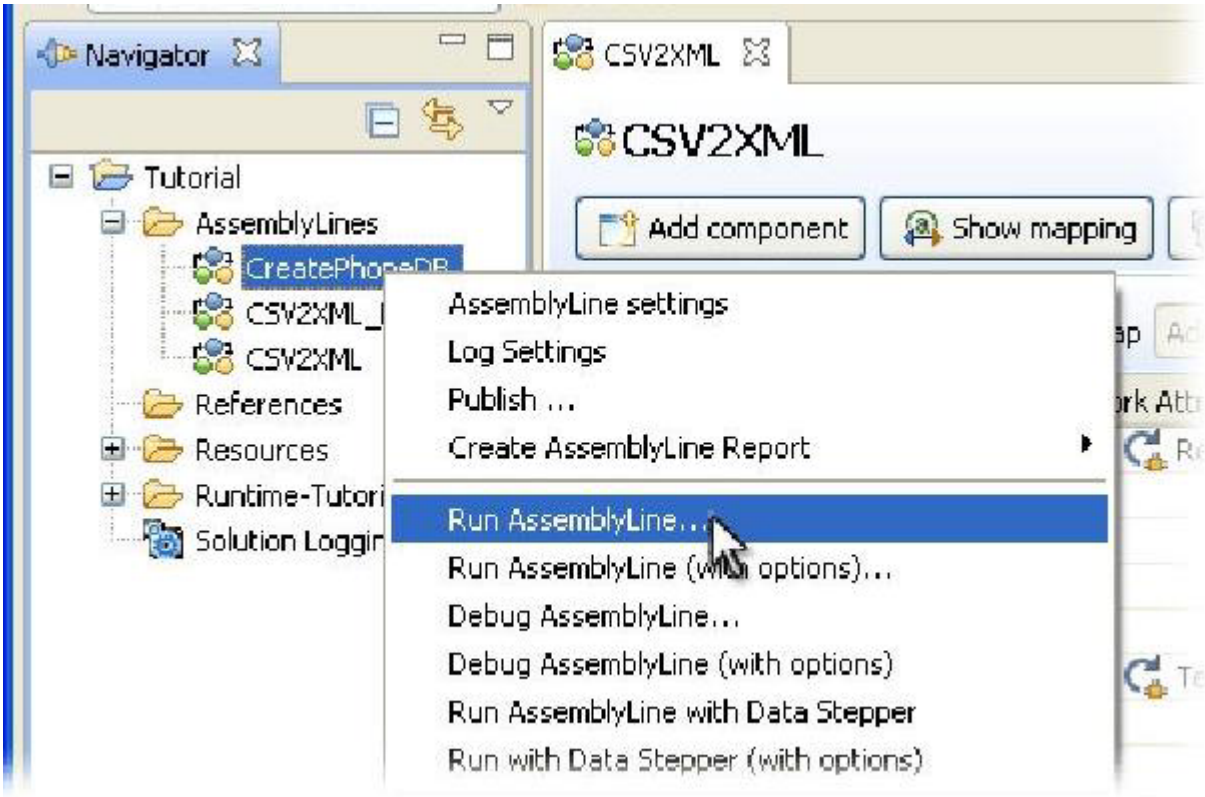

図 67. CreatePhoneDB AL の実行

この AssemblyLine は、まず Derby[27] データベースを「TutorialDB」というソリューション・ディレクトリーの下に作成し、「PhoneDB」テーブルをセットアップします。そして PhoneNumbers.xml 内をループして、その情報を新規テーブルにロードします[28]。

すべてが順調な場合、ログ出力は次のようになります。

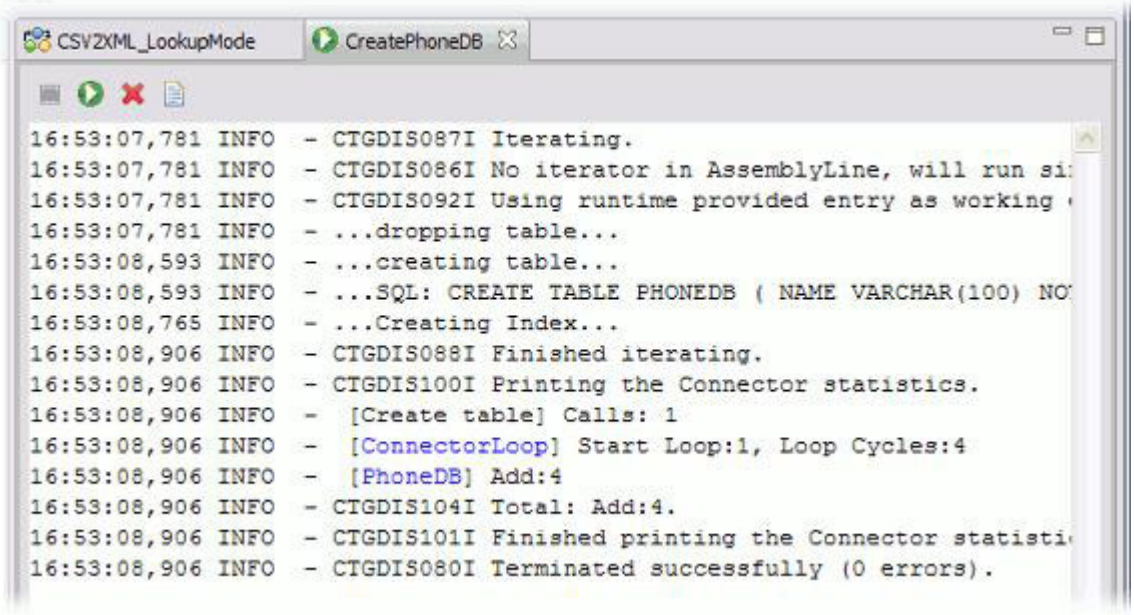

図 68. 「CreatePhoneDB」AssemblyLine からのログ出力

「CreatePhoneDB」AssemblyLine に構成済みの使用可能な JDBC コネクターがあります。このコンポーネントをプロジェクト・リソース・ライブラリー (具体的には「コネクター」フォルダー) にコピーして、AL で再利用します。

「CreatePhoneDB」AssemblyLine を開いて「PhoneDB」というコネクターを選択し、ナビゲーター・ツリーの「リソース」の下にある「コネクター」フォルダーにドラッグします。

27. Apache Derby は IBM Security Directory Integrator にバンドルされているオープン・ソースのリレーショナル・データベースです。

28. 本書ではこの AL について説明しませんが、これはほとんどのコネクター・インターフェースに見られるデータ・ソースに固有の機能を活用するために使用される高度なスクリプト作成技法の好例です。この AssemblyLine を、自由に試してみてください。

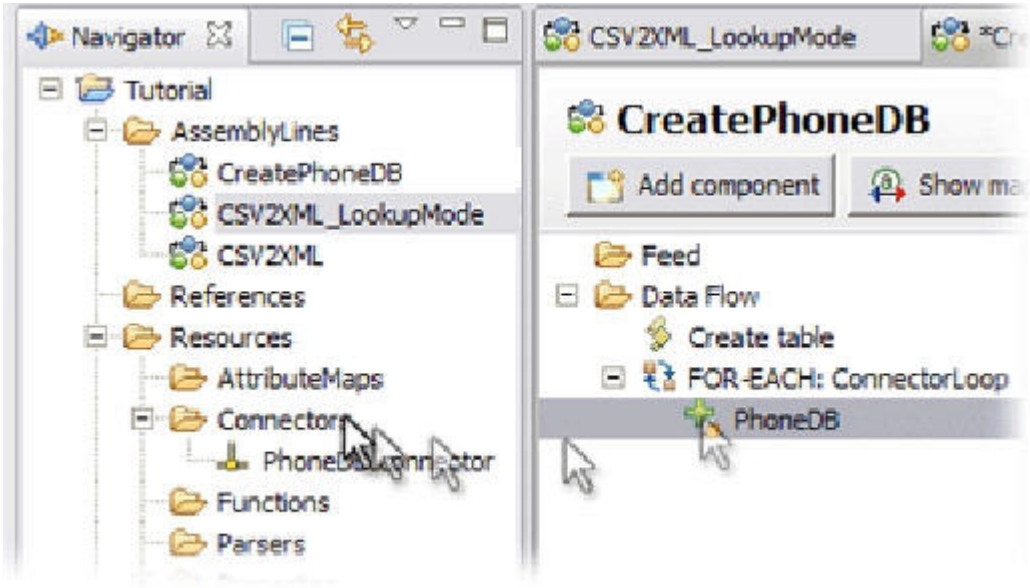

図 69. リソースへのコネクターのドラッグ

「CreatePhoneDB」を閉じて、AssemblyLine が再表示されるようにします。次に、新規の「PhoneDB.connector」リソースを、それまでコネクター・ループが占めていた位置にドラッグします。

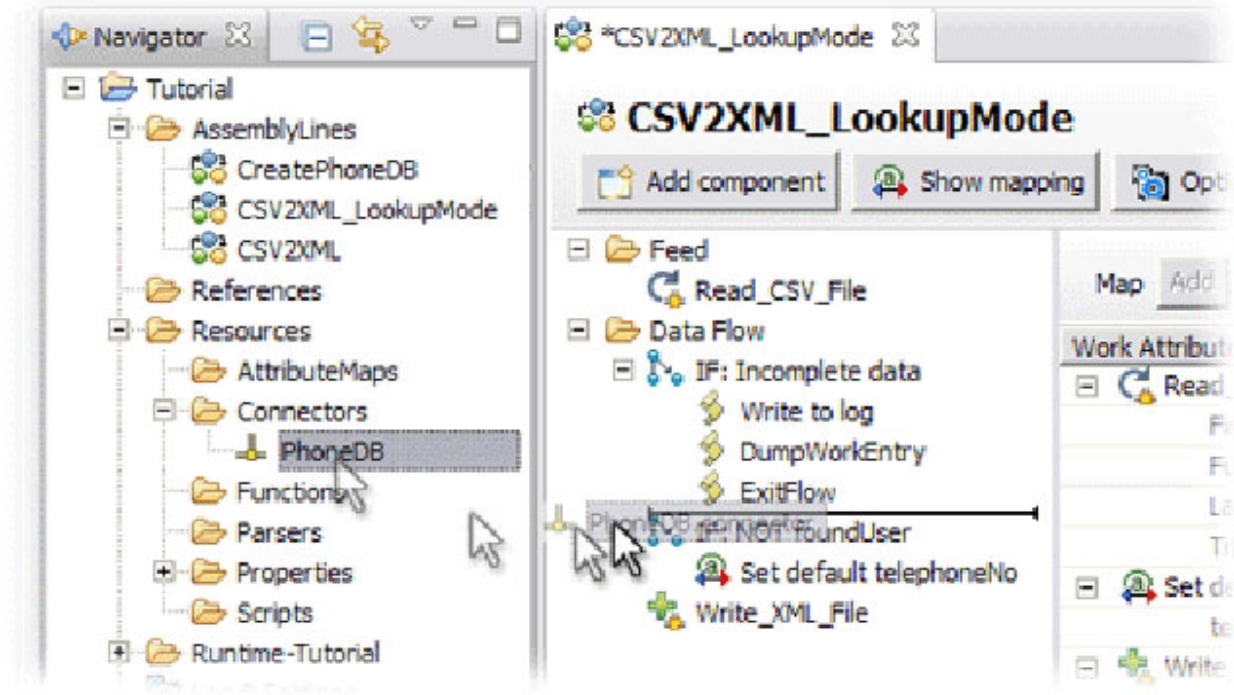

図 70. AssemblyLine への新規リソースのドラッグ

継承

AssemblyLine でこのコネクターが青色で表示されることにご注意ください。これはリソース・ライブラリーから継承しているからです。これはドラッグしたコネクターからランタイムで構成設定を動的に検索することを意味します。この継承機能によって、複数の AssemblyLine で、構成されたコンポーネントやスクリプト化された論理などのリソースを再利用することが容易になります。

エディター・パネルの上部にある「**継承元**」ボタンで上位のコンポーネントを変更することができます。 「構成」、「デルタ」、「入出力マップ」および「フック」のようなコンポーネント・タブにも継承オプションがあり、コンポーネント自体の上位と異なる上位を設定できます。

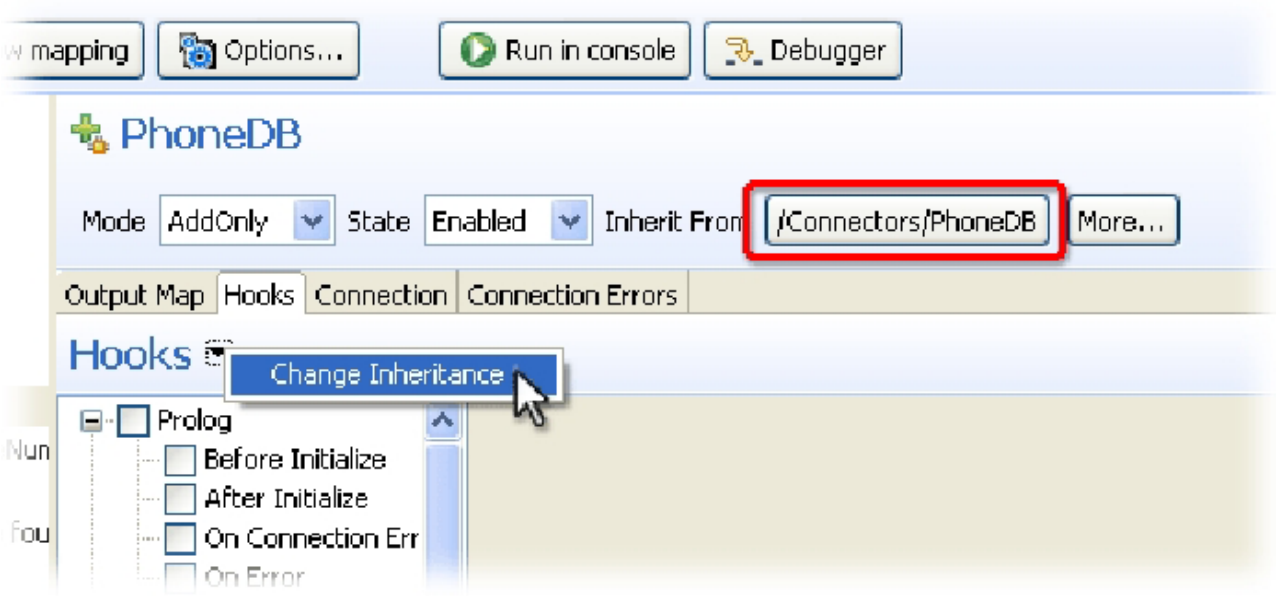

図 71. 「フック」タブの継承の設定

継承値は青色タイプで表示され、それを変更すると継承が破壊されます。継承は、属性マップのルールおよびフックのコンテキスト・メニューにある「**継承値に戻る**」オプションを使用して復元できます。

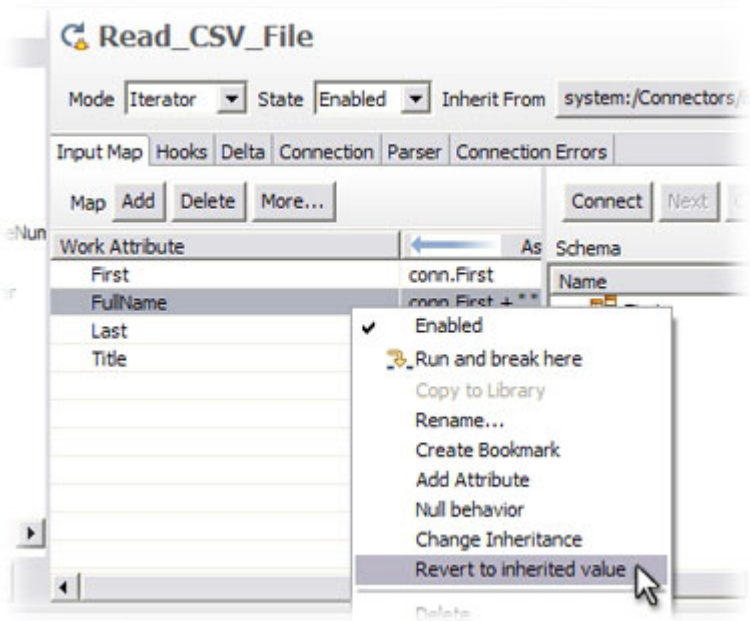

図 72. マッピング・ルールにおける継承の復元

再び練習に戻ります。まず新規の「PhoneDB」コネクターの**モード**設定を「**AddOnly**」から「**ルックアップ**」に変更します。

次に、属性マップが元々は以前のモードに関連付けられている*出力マップ* であったことから、コネクター・スキーマの上にある「**接続**」および「**次へ**」ボタンを押して入力スキーマをディスカバーする必要があります。3 つめ、そして最終ステップは、「PHONE」属性を「スキーマ」から「入力マップ」へとドラッグし、この値の単純マップを得るステップです。

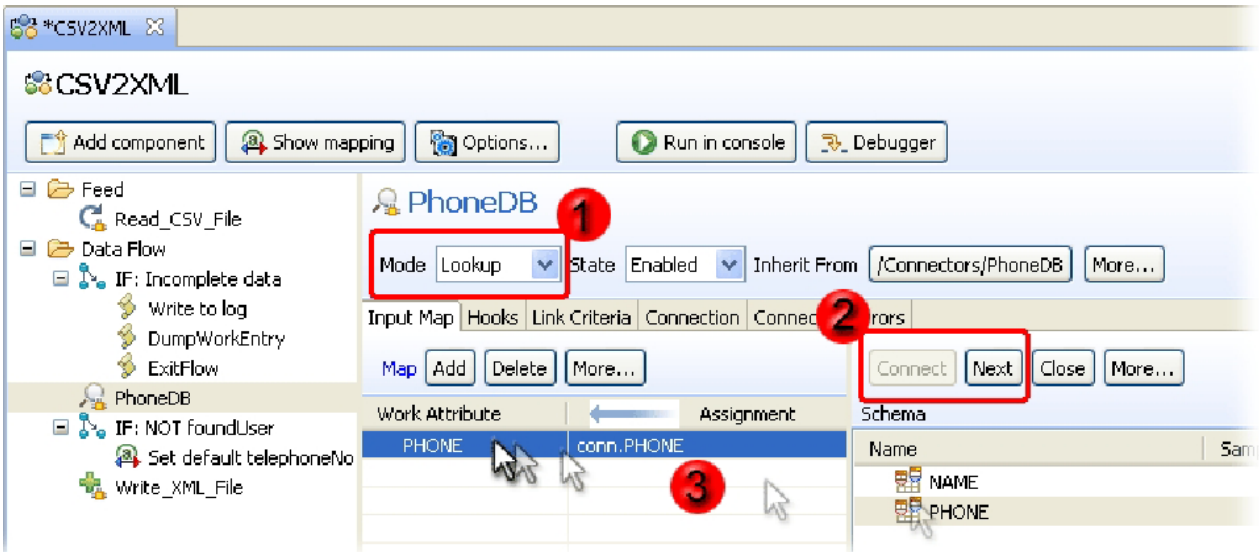

図 73. モードの変更、ディスカバーおよび属性のマッピング

入力マップのルールが確定すると、名前を「PHONE」から「telephoneNo」に変更して、「Write_XML_File」コネクターの「出力マップ」の割り当てに合わせることが重要になります。

「NAME」をマップする必要はないかと疑問に感じているかもしれませんが、ユーザー名は既にあるため、「NAME」をマップする必要はありません。その代わりにこのフィールドは、検索ルールを定義するときに使用します。

ルックアップ検索ルール = リンク基準

ルックアップ・モードを選択すると、コネクター・エディターの中に「リンク基準」というラベルの付いた新規のタブが現れます。「リンク基準」は、検索時の照合ルールを定義するために使用します。

検索ルールは、単純なリンク基準として定義することができます。この場合は、ドロップダウンを使用して、必要に応じて新規のリンク基準を追加し、「条件」の場合と同様に「**いずれかと一致**」チェック・ボックスを設定することによって行います。もう一つの方法としては、「**カスタム・スクリプトで基準を作成**」チェック・ボックスを選択して、次の LDAP 検索フィルターの例のように実際に使う検索ルールを計算するスニペットを記述することができます。

```
"(cn=" + work.FullName + ")"
```

この方法では、接続先のシステムが予期する実際の構文を記述する必要があることから、ソリューションと検索対象のデータ・ソースがより緊密に結び付けられることに注意してください。ここでは、「WHERE」キーワード自体は使用せずに、WHERE 節を作成することを意味します。

一方、単純なリンク基準はコネクターによってネイティブの検索構文に翻訳されるため、リンク基準を再設定しなくてもコネクター・インターフェースを切り替えることができます。

単純なリンク基準は、条件と同じように見えます。最初のドロップダウンには、ユーザーがディスカバーしたスキーマが取り込まれています。2 つめのドロップダウンには、その時 AL 内で使用できる作業項目の属性が表示されます。条件の場合と同様に、ドル記号は、それが付いた名前の属性の値が、検索フィルターを作成するために実行時に置換されることを示しています。

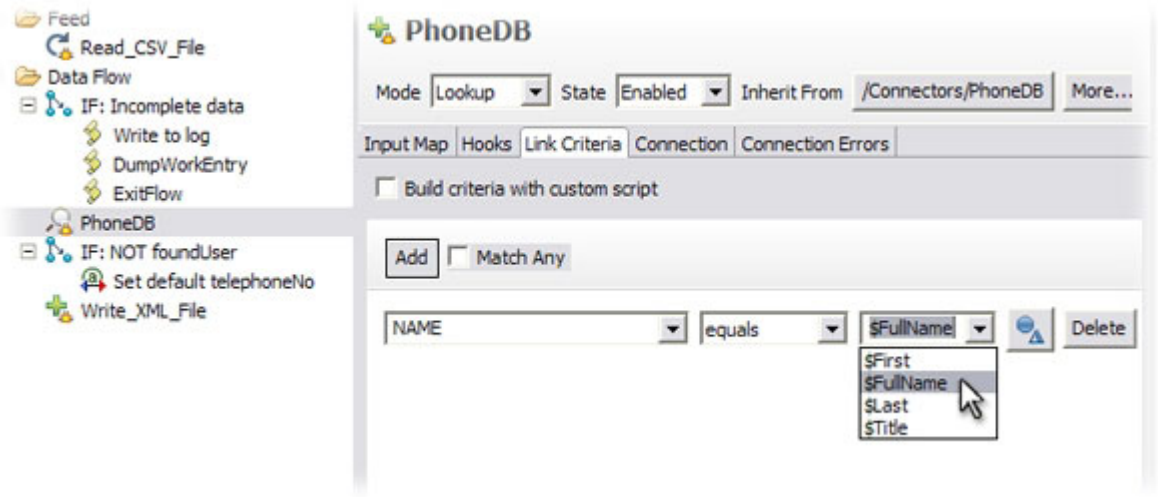

図 74. 単純なリンク基準

CTRL-S を押すなどして、作業を保存するのを忘れないでください[29]。 定期的に保存を行えば、作業を喪失することは避けられます。

ここで再び、AssemblyLine を「**実行**」します。

実行時エラーの解読

冷静になりましょう。ログ出力に表示されているスタック・ダンプが、エラーを指し示しているはずです。一番初めのスタック・ダンプの先頭までスクロールします。エラーが発生した*原因* と問題発生*箇所* の両方の情報が表示されています。

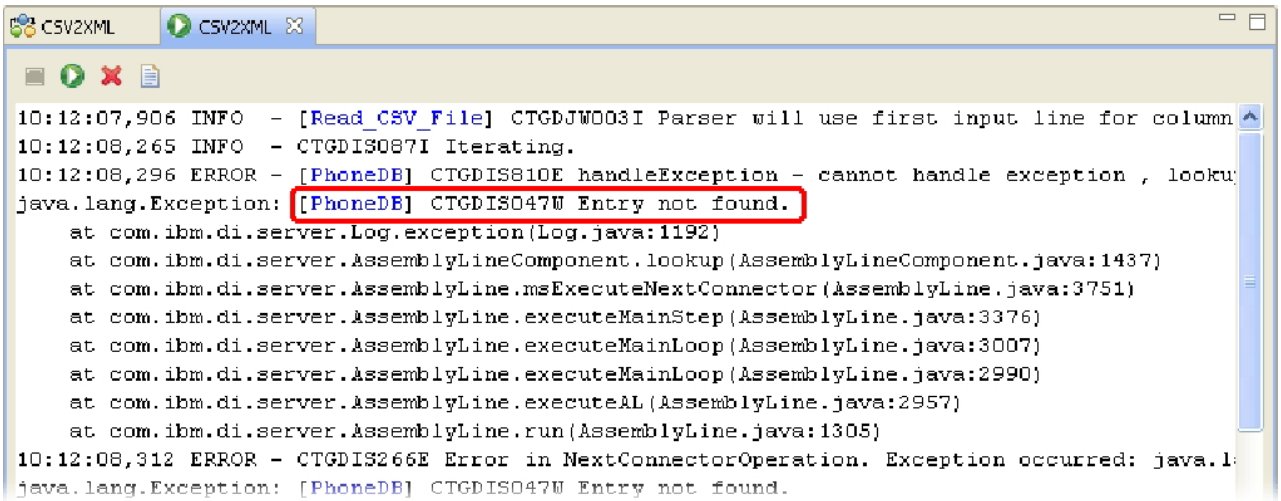

図 75. ログ出力中のエラー・メッセージ

29. AssemblyLine やリソースを削除したあとでそれを元に戻す必要が生じた場合、ナビゲーター・パネルにあるプロジェクトを右クリックし、「**ローカル履歴からリストア...**」を選択すると、そこからリストア可能な資産バージョンのリストが表示されます。ただし、当然のことですが、事前に何も保存されていなければ、ローカル履歴にも表示されません。

コンポーネント名が大括弧で囲まれ (「PhoneDB」)、項目が見つかりません、つまり検索が失敗しましたというエラー記述が続いています。

コネクターがルックアップ・モードで構成されているとき、システムは検索の実行時に 1 つが検出されること、しかも、ただ一つのレコードがマッチングすることを期待します。何も検索されないか、複数のレコードがリンク基準に一致した場合には、AL の停止を防止するために、少なくとも使用可能にする必要のある特別のフック で検索が終了します。この動作は、「*リファレンス*」に記載されているデータ・フロー・ダイアグラムで明確に説明されています。次の図は、ルックアップ・モードの詳細が記述されているページから抜粋されたものです。

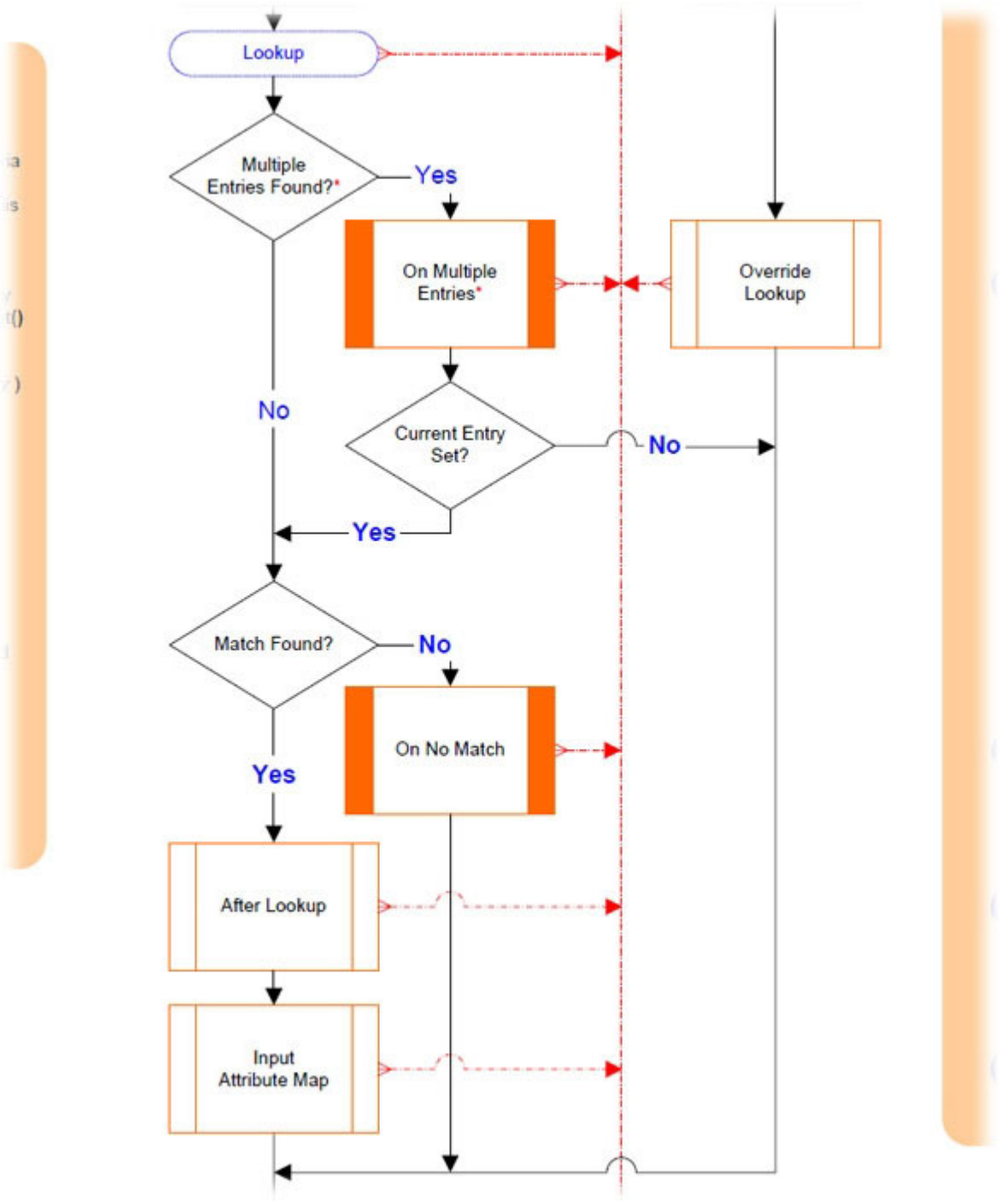

図 76. 「ルックアップ」モード用フロー・ダイアグラムの一部

この動作を活用して、foundUser フラグ変数を設定することができます。「PhoneDB」コネクターを右クリックして「**フック**」を選択し、フック・エディターをオープンします。「一致なしの場合」フックを選択して foundUser を *false* に設定するスクリプトを入力します。

```
foundUser = false;
```

「ルックアップ後」を選択してこの相補的なフック・スクリプトを入力します。[30]

```
foundUser = true;
```

AssemblyLine の表示は、スクリーン・ショットに示した例の一部のようになります。

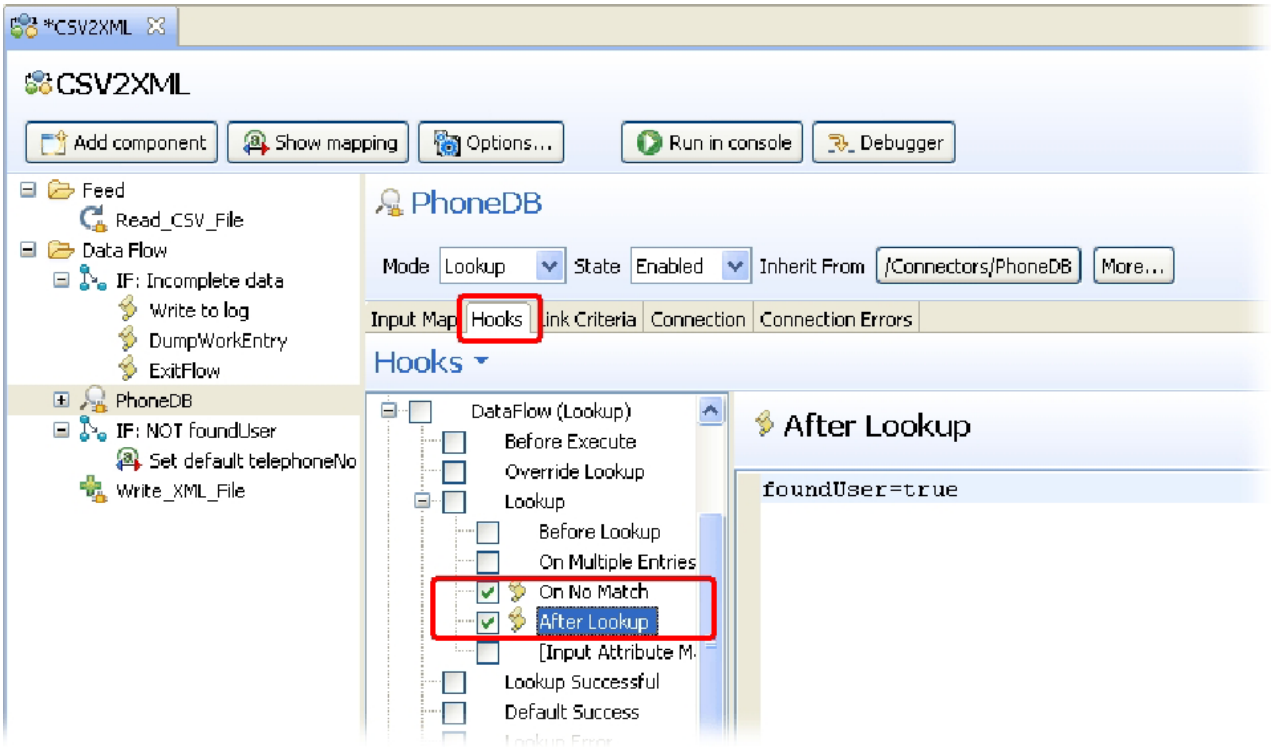

図 77. 最初のチュートリアル練習の終了

この AssemblyLine を再び**実行**すると、今度はエラーなしで終了するはずです。`Output.xml` ファイルは Loop ベースの AL からの結果と同一になっていることを確認してください。

お疲れ様でした。最初の IBM Security Directory Integrator チュートリアル練習が終了しました。次は、AssemblyLine をリアルタイム・イベントで始動させる方法について記載します。

30. このフロー・ダイアグラムから、「ルックアップ後」フックは検索結果が唯一の一致であった場合にのみ実行されることがわかります。

第 3 章 イベント・ドリブンによる統合

これまで AssemblyLine をバッチ・プロセスとして実行し、データを流す度に手動で起動させていました。次のように IBM Security Directory Integrator サーバーを起動させることで、コマンド行から AssemblyLine を実行することもできます[31]。

```
ibmdisrv -c examples/Tutorial/Tutorial1.xml -r CSVtoXML
```

こうして、処理をスケジューリングするためにスケジューリング・ツール (*crontab* など) を使用することや、外部アプリケーションからスケジューリング・ツールを起動することは、簡単なタスクです。

IBM Security Directory Integrator は AssemblyLine からイベントを検知できるようにして、広範囲にわたる様々なリアルタイム・トリガーをソリューションが処理して応答できるようにする数多くの機能を提供します。

こうしたトリガーの例を以下に示します。

- REST コール、SNMP や Web サービス等、IP ポート経由のプロトコル・リクエスト
- キュー上に出現する新規メッセージ
- Inbox に到着する電子メール
- ファイル、データベース、ディレクトリー 、IBM Notes データベースなどのデータの変更
- スケジューリングまたはタイマー・ベースの AssemblyLine 操作

これがすべてではありません。他の IBM Security Directory Integrator 文書や、コミュニティ Web サイト、ニュースグループでも、着想や指針を得られるでしょう。

ソリューションでのこれらのイベント処理は、多くの方法で実施できます。

イテレーター・モードのコネクター

コネクターによっては、イテレーター・モードの場合にタイムアウト・パラメーターを構成することができます。ファイル・システム・コネクターはその一例で、ファイルを最後まで読み出してから、新たな情報が現れるのを待機するように設定することができます。これは「末尾読み出し」と呼ばれます。

RDBMS 変更検出や LDAP 変更ログのコネクターを含む他のコネクターは、同様の方式で動作します。これらのコネクターを使用すると、接続されたシステムに新たな変更が現れるのを待機しながら連続的に実行される AssemblyLine を構築できます。

またイテレーター・モードで実行され、AssemblyLine をスケジューリング・パラメーターによるインターバル・タイマーで動作するように構成できるタイマー・コネクターもあります。これは手軽にテストができます。

31. コマンド行引数なしで `ibmdisrv` で起動された場合には、IBM Security Directory Integrator Server は使用方法のメッセージを表示します。

IBM Security Directory Integrator の標準インストール構成には Web 管理コンソールが含まれています。この Administration and Monitoring Console 管理およびモニター・コンソール (AMC) というブラウザー・ベースのアプリケーションを使用して、 AssemblyLine のヘルス・モニター、実行中のサーバーへの構成のロード、AL の起動と停止、統合ソリューションの高可用性を保持するための障害/応答動作の構成を行うことができます。これは、AssemblyLine が起動する必要のある時間のスケジュール設定に使用することもできます。しかし、この Web 管理ツールについては本書では扱いません。

注: AMC 機能は推奨されず、IBM Security Directory Integrator の将来のバージョンで除去されます。

サーバー・モードのコネクター
HTTP Server コネクターや LDAP サーバー・コネクターといった少数の特殊なコネクターを使用すると、外部クライアントから来る受信要求を処理して、要求動作を実行し、適切な応答を返すソリューションを構築することができます。次の練習では HTTP Server コネクターを使います。

通知およびプロパティー
これらのイベント送信用のコンポーネント (およびスクリプト呼び出し) があるために、IBM Security Directory Integrator は異なるプラットフォーム上で動いている別々の IBM Security Directory Integrator サーバー間の IBM Security Directory Integrator 通知イベントにさえサブスクライブできるコンポーネントを持っているのです。

本書では、AL のスケジューリングとコネクター・サーバー・モードについて詳しく解説します。

AssemblyLine のスケジューリング

構成エディターのユーザー・インターフェースを使用して、指定した時刻に AssemblyLine を実行するスケジューラーを作成できます。例えば、毎日午前 3 時 05 分に、土曜日の午前 7 時に、またはさらに複雑なスケジュールで実行するスケジューラーを作成できます。

スケジューラーを作成するには、構成エディターで「**ファイル**」 > 「**新規作成**」 > 「**スケジューラー**」をクリックします。別の方法では、AssemblyLine を右クリックし、「**スケジュールの作成**」を選択します。

IBM Security Directory Integrator サーバーが構成ファイルをロードすると、その構成ファイルに格納されている使用可能なスケジューラーが自動的に開始されます。構成ファイルをロードするコマンドは、`ibmdisrv -c myconfig.xml -d` です。ここで `myconfig.xml` はエクスポートされた構成ファイルです。構成インスタンスを停止すると、関連付けられているすべてのスケジューラーが停止します。

図 78. IBM Security Directory Integrator スケジューラー

IBM Security Directory Integrator スケジューラーの詳細は、「リファレンス」の『*IBM Security Directory Integrator スケジューラー*』のセクションで詳述されています。

サービス要求 AssemblyLine

この最後の練習では、「CSV2XML_LookupMode」AL つまり、データ転送を開始するための HTTP サービスを起動する非常に単純なユーザー・インターフェースを備えた Web サーバー AssemblyLine を構築します。

まず、新規の AL を作成して、「TINA_WebServer」という名前を付けます[32]。次に、新規コンポーネントを挿入して、「HTTP Server コネクター」を選択し、「**完**

32. TINA は「TINA Is Not Apache」の省略です。

了」を押してウィザードを終了します。そして、「**入力マップ**」タブを開きます。

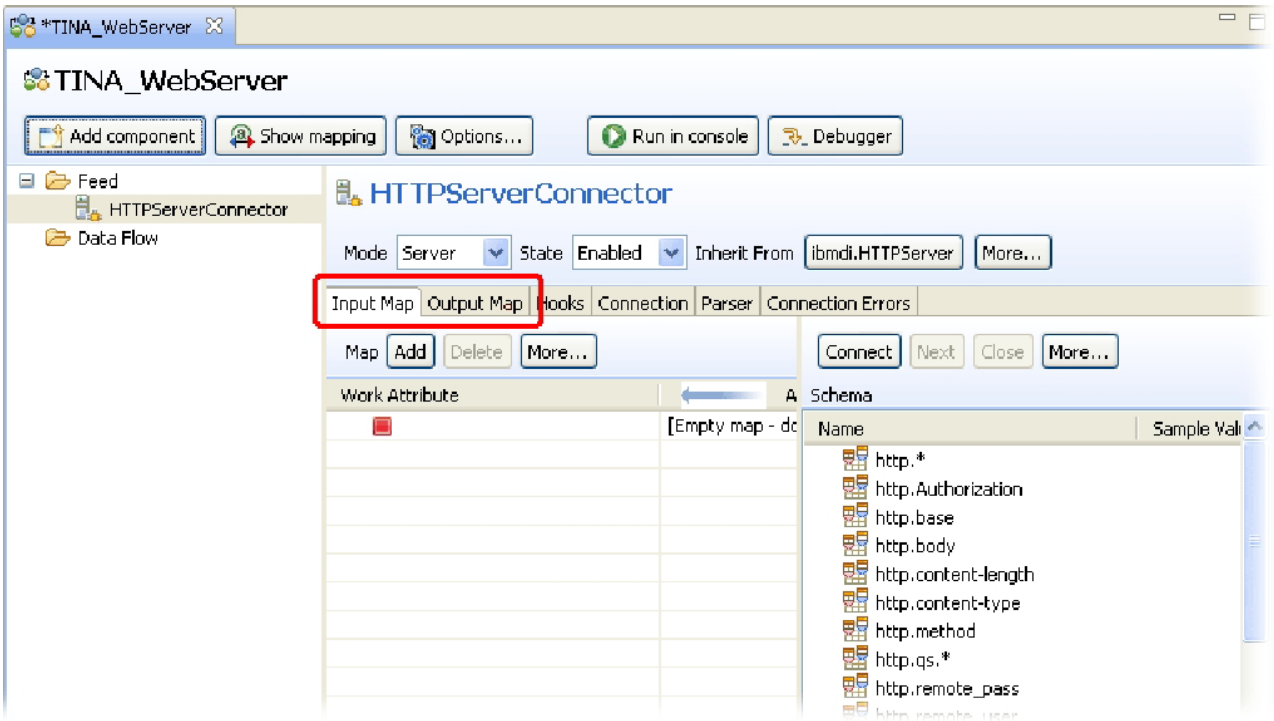

図 79. HTTP Server コネクター属性マップ・パネル

サーバー・コネクターは、関数コンポーネントと相補的な関係にあります。FC がサービス要求を行うのに対し、サーバー・コネクターはサービスを提供して支援します。結果として、要求を行うクライアントからの属性を受け取るために、サーバー・コネクターの入力マップが使用されます。一方、出力マップは応答するための方法を提供します。属性マップの画面の右側部分に入力スキーマが表示されています。これは、出力スキーマについても同様です。サーバー・コネクターは、マッピングを行うために有用な情報を提供します。スキーマ属性には、「http.qs.*」などのワイルドカード文字が含まれる場合があることに注意してください。これは、名前が「http.qs.」で始まる着信属性の番号を知らせるための設計時の情報です[33]。

最後に、すべてのサーバー・コネクターの「**モード**」ドロップダウンは、サーバー・モードとイテレーター・モードの両方を提供します。ここには表示されていませんが、もう 1 つモード (応答モード) があり、合計で 3 つのモードがあります。サーバー・コネクターは、操作中のさまざまな段階でモードを切り替えます。

1. サーバー・コネクターは、まずサーバー・モードで開始し、IP ポートなどのリソースに接続して、クライアント接続の着信を待ちます。
2. 接続が確立すると、コネクターはイテレーター・モードに切り替わって、入力マップに基づいてクライアント・データを検索し、それをデータ・フロー・コンポーネントに渡します。
3. 最後に、データ・フロー・コンポーネントが実行を完了すると、コネクターは応答モードになり、出力マップを使用してクライアントに返す応答を作成します。

33. この属性 (http.qs.*) の特別な集合は、クライアントによる HTTP の呼び出しに渡される照会ストリングのパラメーターを保持しています。

これらは自動的に処理されるので、操作する必要はありません。ただし、フックのスクリプトを記述する場合には、サーバー、イテレーター、および応答 の 3 つのフックがあることに注意してください。

続いてのチュートリアルの練習では、入力マップ属性の項目を追加します。

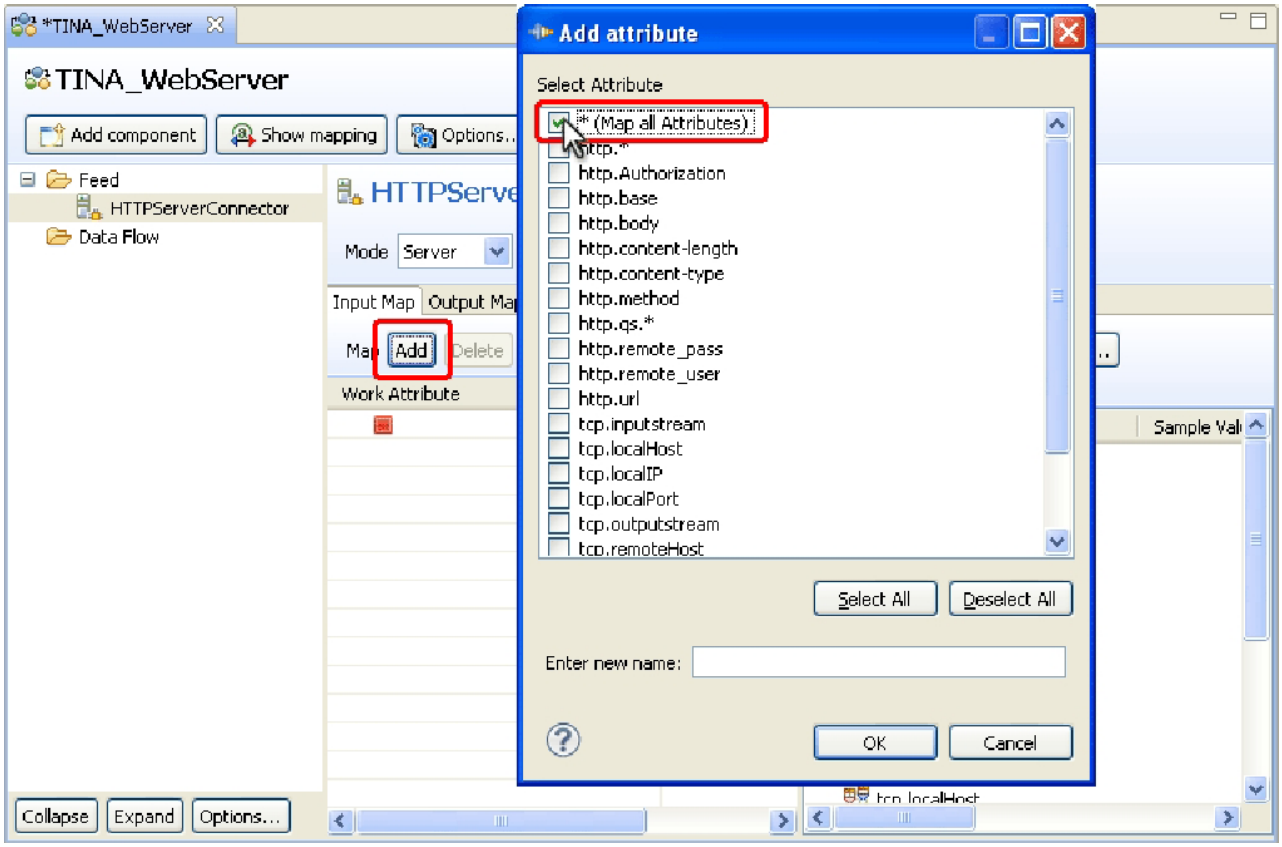

図 80. 新規入力属性マップの追加

表示されたダイアログの一番上のオプションを選択します。または、「Entry new name」フィールドでアスタリスク文字 (*) を入力することもできます。これは、特別なワイルドカード・マップ・ルールで、これを指定すると、コネクターによって読み取られるすべての属性に IBM Security Directory Integrator がマップされます。

入力マップは、このように表示されます。

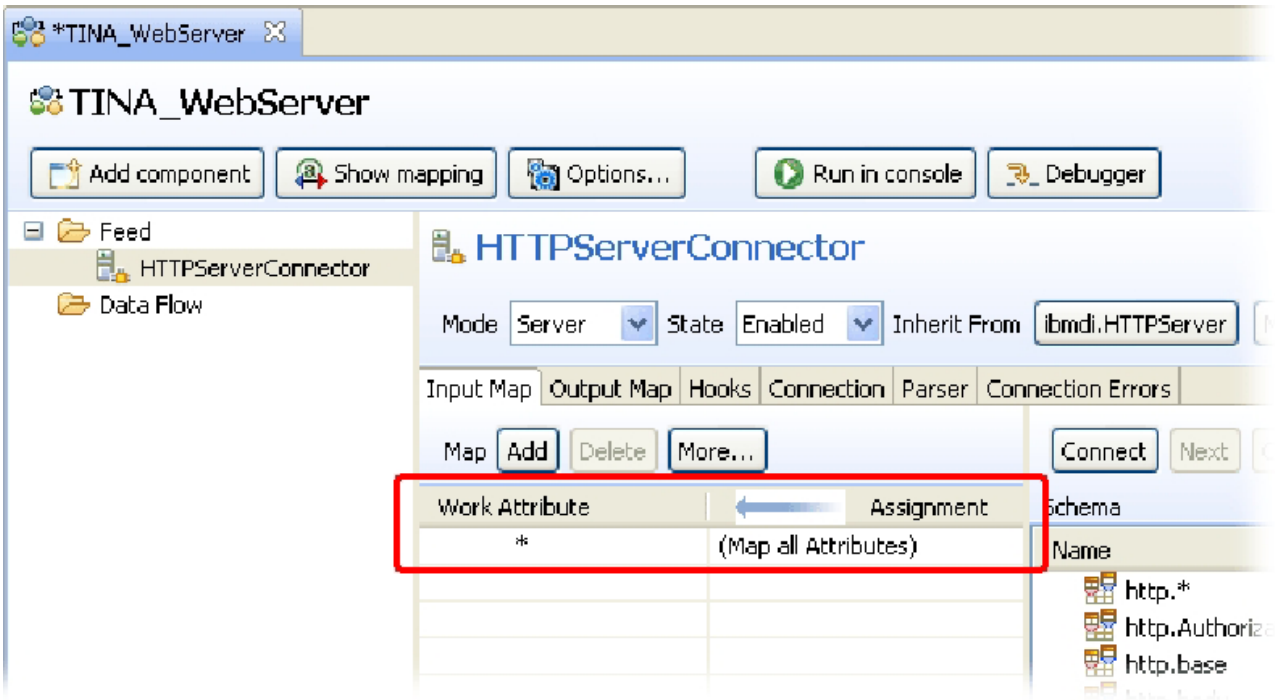

図 81. ワイルドカード・マップの項目

ワイルドカード・マップの項目を出力マップにも追加して、応答メッセージの作業項目に設定された属性のすべてがクライアントにマップされるようにします。

このコンポーネントをテストするには、「作業項目のダンプ」のスクリプト・コンポーネントをフロー・セクションに配置して AssemblyLine を「**実行**」します。ログ出力には、HTTP Server コネクターがポート 80 [34]で HTTP コネクションをリスニングしているメッセージが表示されます。これは、AL がクライアントの接続を待機中であることを示しています。ブラウザーを開いて次の URL にナビゲートしてください。

```
http://localhost:80
```

ログの出力を参照してください。属性として戻された多数の TCP や HTTP のヘッダー・プロパティーが表示されています。

34. 何らかの理由によりこのポートがすでに使用中の場合は、HTTP Server コネクターの構成パネルを開いて他のポートを選択してください。

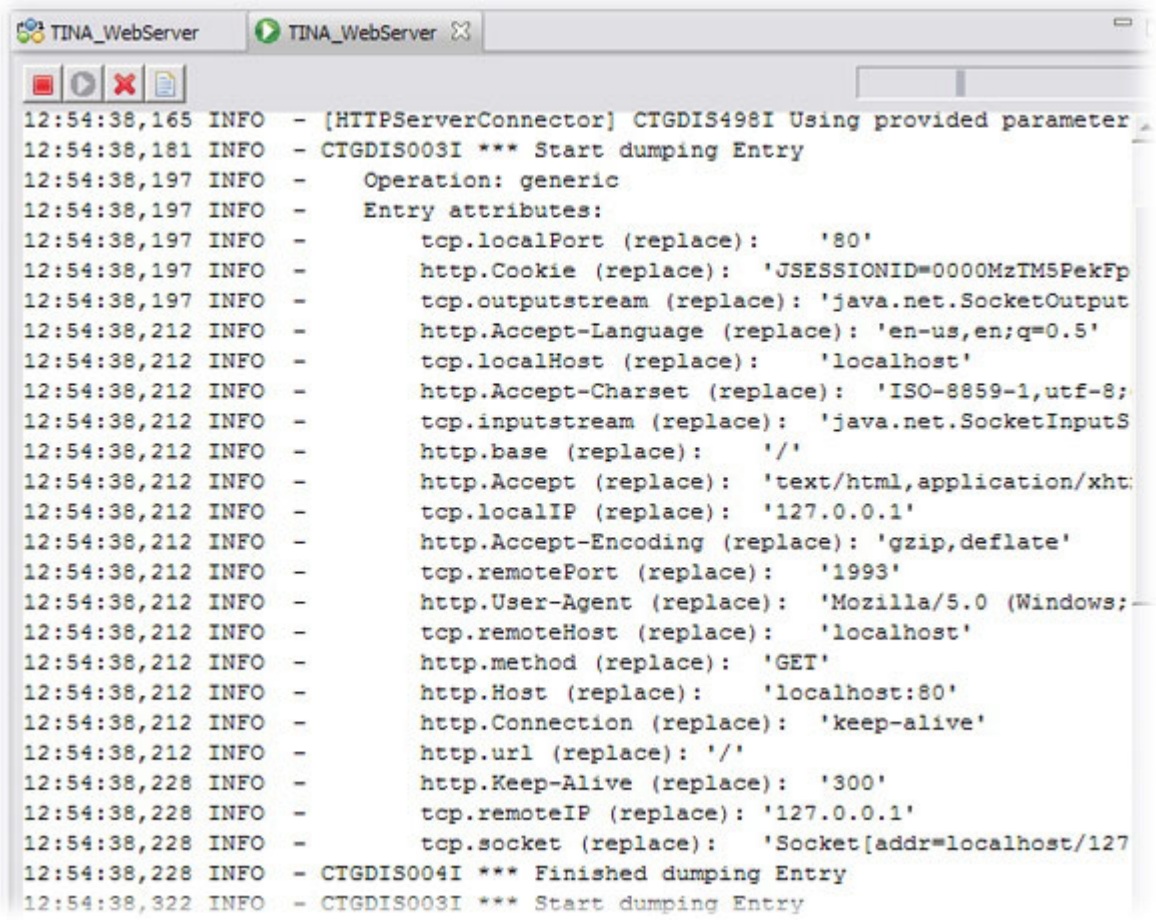

図 82. 属性として戻された TCP や HTTP ヘッダー・プロパティー

今回の練習に関係のある属性は、ホストとソケットの後に表示されている URL の一部を伴う「http.base」のみです。具体的には、「RunAL」というテキストを含むものが対象です。

このテキストを確認するには、AL のデータ・フロー・セクションに IF ブランチを追加します。それに、「RunAL detected」という名前を付けて、条件に http.base *contains*「RunAL」を設定してください。

このブランチ条件の評価が *true* の場合、「CSV2XML_LookupMode」AL を起動するようにします。そのためには、まず、「Scheduler」AL の AssemblyLine 関数コンポーネントをナビゲーター・パネルの「**リソース**」 > 「**関数**」にドラッグしてから AssemblyLine に戻り、新規 IF ブランチの上にドロップして再利用します。

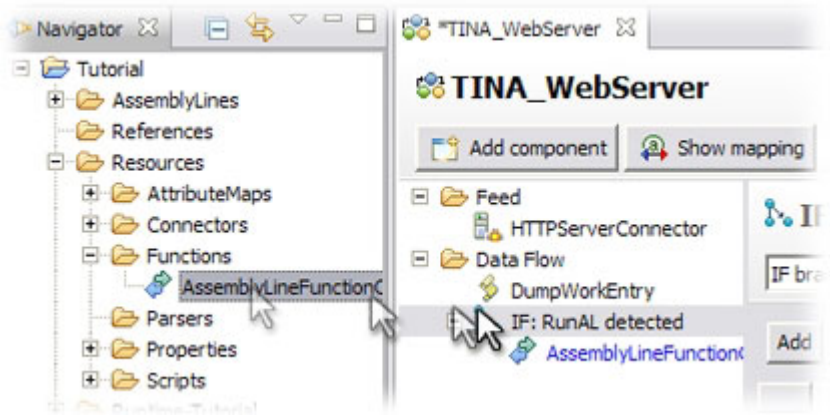

図 83. AssemblyLine 関数コンポーネント (AL FC) のドラッグ

次に、AssemblyLine を再実行して、次のテキストをブラウザーのアドレス・フィールドに入力します。

```
http://localhost/RunAL
```

作業項目のダンプとともに、呼び出された AssemblyLine の統計情報がログ出力に表示されます。

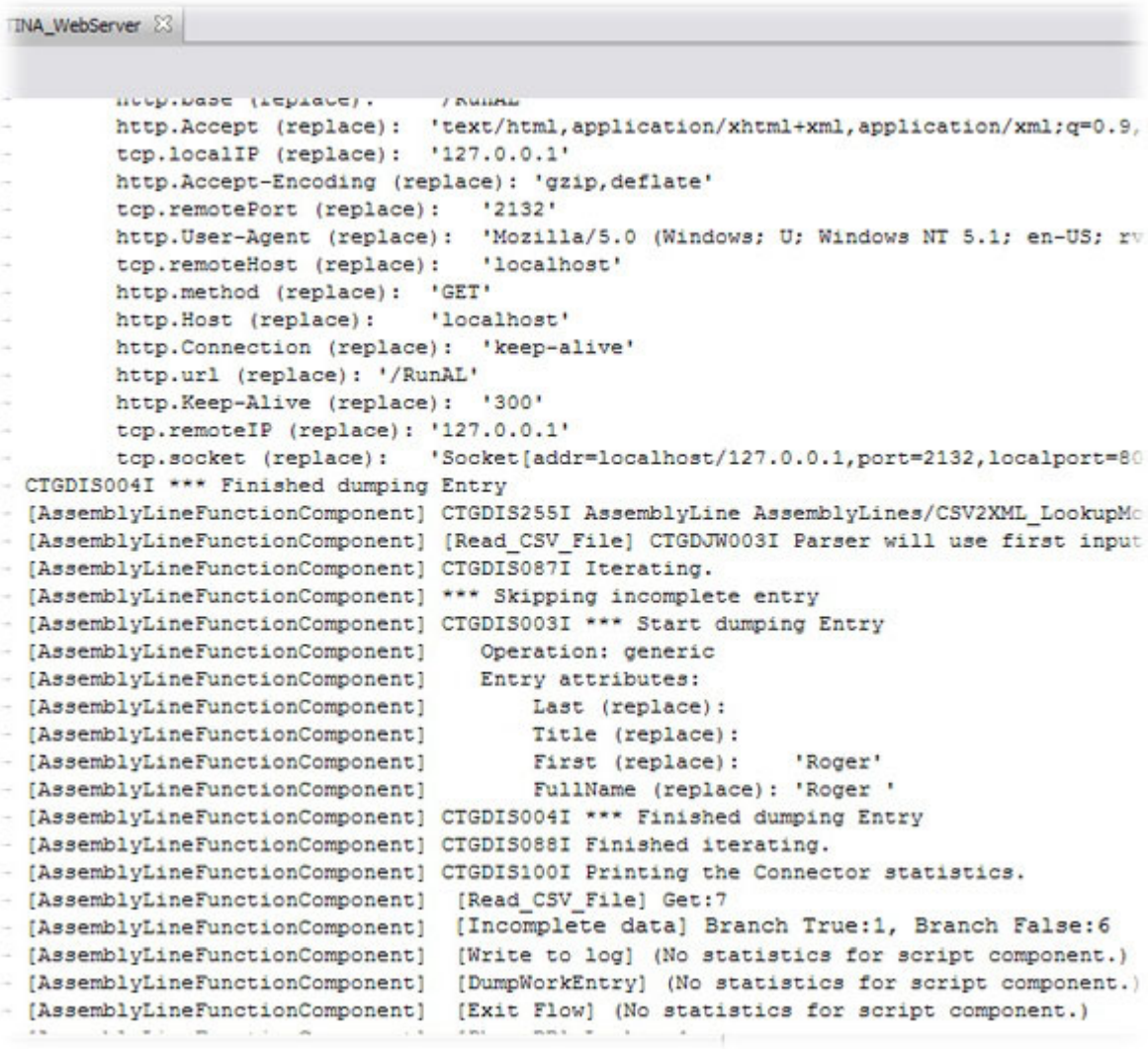

図 84. 作業項目のダンプと AL 統計情報

これでサービスが動作するようになりますが、まだ練習は終わりではありません。まず、 Web サーバーに HTML ページを返させることによって、よりすっきりと使いやすいものにすることができます。そのためには、通常、多少のスクリプト記述が必要になります。幸い、ドラッグ・アンド・ドロップするだけでこれが可能になるチュートリアル用のファイルが複数用意されています。

始める前に、「作業項目のダンプ」スクリプトを使用不可にしてログへの出力を最小限にしてください。次に、ELSE ブランチを「IF RunAL detected」のすぐ後ろに追加します。それに「Return web page」という名前を付けます。ファイル・ブラウザーを使って、Tutorial ディレクトリーの `Return web page.script` という名前のスクリプト・ファイルを確認し、それを「**リソース**」 > 「**スクリプト**」にドラッグします。そこから AL に引っ張って、ELSE ブランチの上にドロップします。AL は、次のように表示されます。

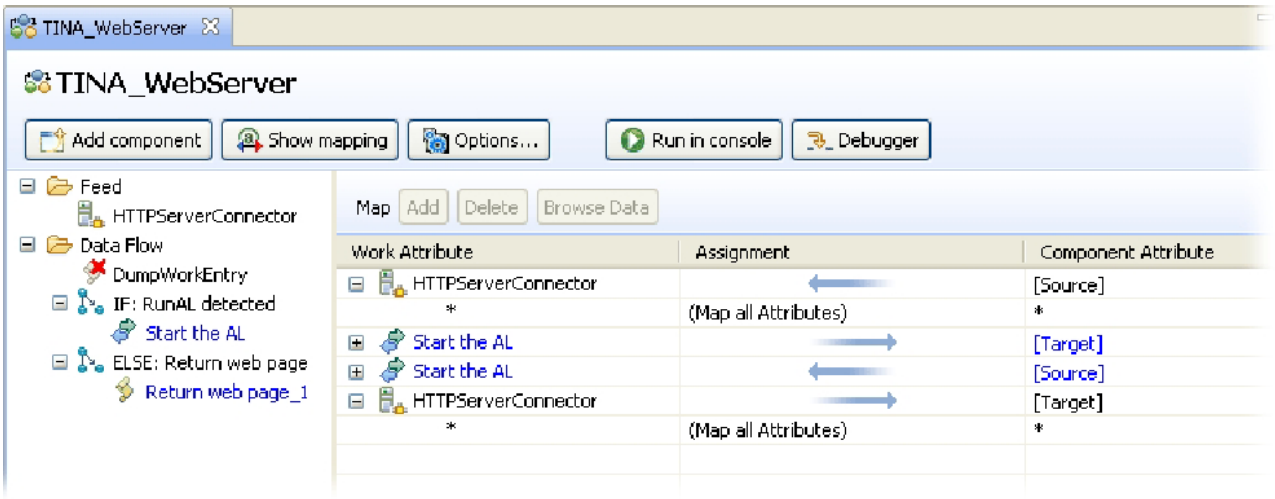

図 85. 完成した TINA_WebServer AssemblyLine

もう一度 AssemblyLine を実行して http://localhost を呼び出すと、Web ページの表示はこのようになります。

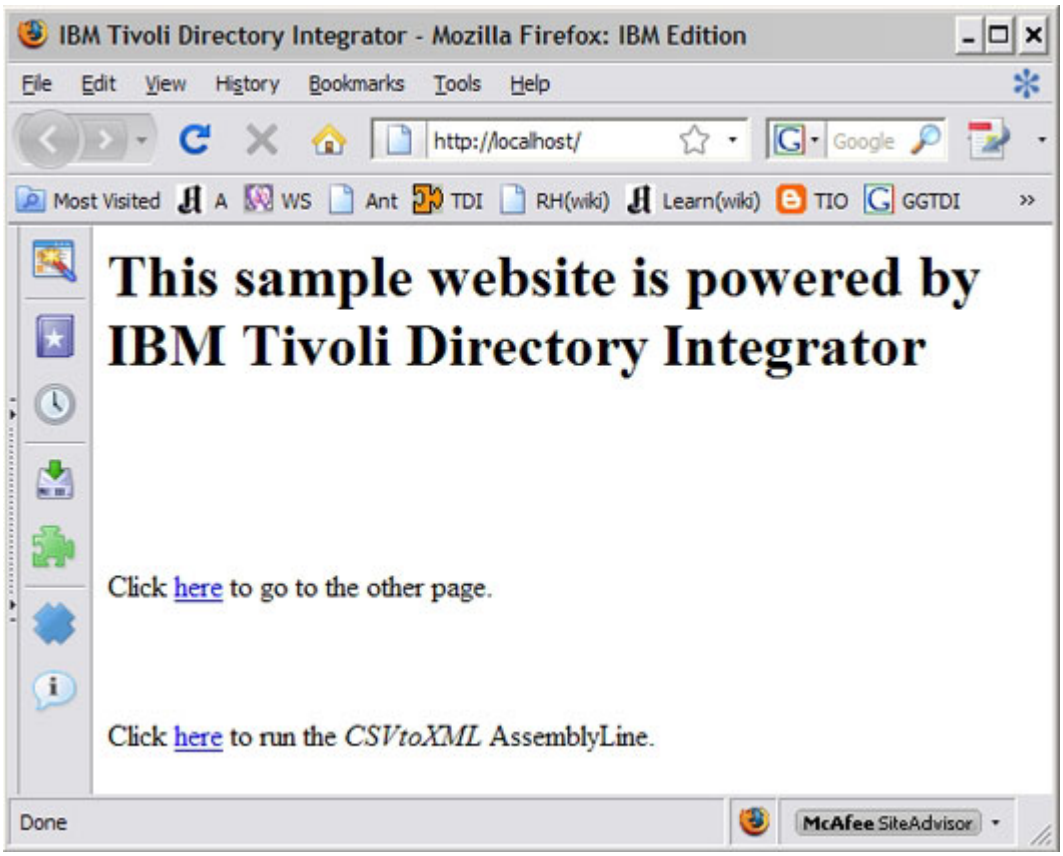

図 86. ソリューションの単純な Web インターフェース

一番上のリンクは `OtherPage.html` へナビゲートします。一方、下部のリンクは、AssemblyLine を起動するために、要求された http.base の「RunAL」テキストを送信します。

本書の実践はここまでです。

第 4 章 統合ソリューションの強化

ここまで見てきたように、AssemblyLine はかなり手早く作成できます。ただし、AL の実行が完了したからといって、必要な時に向けてソリューションの準備が整ったわけではありません。単純な統合タスクでさえ、最低限の事前計画と成果の分析を保証するのを当然としています。

以下のような質問について、よく考えてみる必要があります。

- すべてのソース・データが期待どおりに処理されていますか。それをどのようにして確認できますか。
- データの内容または品質、あるいはその両方に異常が検出されませんでしたか。それらの異常に対処できていますか。
- 処理が他のデータ・セット、システム、または AL に影響を与えていますか。それはどのように影響を与えていますか。
- 統合に伴って監査の責任が生じていますか。
- ソリューションを配置するのは誰ですか。それを使用しているのは誰ですか。また、管理作業をするのはだれでどのように行いますか。

同期化のために使用されるもの、電力供給のために使用されるものなど、長期間にわたって実行される AssemblyLine については、さらに可用性、耐障害性、パフォーマンス、スケーラビリティー、およびセキュリティーなどについての考察を付け加えることができます。

この最終セクションの目的は、これらの問題に加え、それに対処するために使用できる IBM Security Directory Integrator の複数の機能および技術を認識されることです。

注: IBM Security Directory Integrator の新規ユーザーの方は、本セクションが複雑で難解に思われても心配する必要はありません。システムを使い慣れて経験を積んだ後に、この章の各セクションを読み返してください。

可読性、再利用、および構成可能性

すべての開発作業において、トラブルシューティング、保守、および拡張作業は不可欠です。IBM Security Directory Integrator ソリューションも例外ではありません。

以下の基本的なガイドラインを守ることによって、これらを実現できます。

1. AssemblyLine は、他の人々が理解し、使用し、管理する必要があることを念頭に置いて記述します。これは、AL をできる限り短くし、コンポーネント名を明確で説明的なものにするということです。ブランチとループを使用して AL のフローにインプリメントされたロジックの方が、フックに「隠された」ロジックやスクリプト・コンポーネントにパックされたロジックよりも、プログラマーでないユーザーにとっては読みやすく、デバッグも容易です。

2. *短い AL* というルールは、必然的に、スクリプトのスニペットを短くすることにもつながります。一体構造のコードのブロックを記述するのではなく、それらをより小さい単位に分割して、場合によっては別個のスクリプト・コンポーネントに分けて記述すれば、可読性とデバッグのしやすさが向上します。そうすれば、1 つの SC を無効にすることで、コードをスキップできます。

   可読性を改善し、コードの重複を避けるためのもう 1 つの方法は、(ナビゲーターツリー・ビュー内の「スクリプト」フォルダーから) スクリプト・コンポーネントの継承を利用し、共通タスク用に関数を定義することです。AssemblyLine は、独自のスクリプト・エンジンのコンテキストで実行されます。そのため、1 つの場所で宣言された変数および関数のすべてが、あらゆる場所から利用可能になります。これらを定義する共通の場所は、AL のプロローグ・フック、または「追加のプロローグ」として選択されているスクリプトの中にあります[35]。
3. 自分が作ったものでも、6 カ月経てばまるで他人が作ったもののように感じるということを念頭に置き、アルゴリズムは、洗練性よりも可読性を優先しましょう。同僚に対する配慮は、自分自身に対する配慮ともなります。
4. 構成エディターのスキルを持たない人が、事情により設定を変更して AssemblyLine を実行する場合もあるので注意してください。AL は、次のようなコマンド行を使用して簡単に開始できます。

   ```
   ibmdisrv –c myConfig.xml –r myAssemblyLine
   ```

   スクリプトまたはバッチ・ファイルを作成して、これを容易にすることができます。
5. プロパティーを使用してパラメーター設定を外部化することにより、AL の再構成をより簡単にしてください。プロパティーは、ファイルまたはデータベースに保存できる、キー/値ペアです。これを使用して、構成エディターの外部からソリューションを再構成できます。プロパティーは、パラメーター・ラベルをクリックし、「**プロパティーの追加**」ボタンを押すことによって、コンポーネントのパラメーターに結び付けることができます。

   また、プロパティーは、スクリプトから `system.getTDIProperty()` 呼び出しおよび `system.setTDIProperty()` 呼び出しを使用して照会および変更することもできます。そのため、外部プロパティー設定を使用した、カスタム・ロジックの切り替えも容易に行うことができます。

   さらにプロパティーは、コマンド行ユーティリティー `bin/tdisrvctl` を使用して、実行中のサーバー内で変更できます。このユーティリティーでは、すべてサーバーを停止することなく、AssemblyLine の開始および停止、状況の照会、および構成の読み込み (再読み込み) も可能です。
6. 既に述べたように、ファイルに相対パスを使用することで、ソリューションを新規の場所へ移動してインストールするのが容易になります。ファイルのパスは、Config XML ファイルを読み込むディレクトリーに対する相対パスにすることを推奨します。このディレクトリーには、`{config.$directory}` プロパティーを介

35. 追加のプロローグは、AssemblyLine の独自のプロローグ・フックが呼び出される前に実行されます。これらは、AssemblyLine 設定パネルで選択できます。このパネルには、AL を右クリックしてから「**AssemblyLine 設定...**」を選択してアクセスできます。

してアクセスできます。このプロパティーは、例えば次のように、「**Text w/substitution**」オプションを使用して、特定のパス・パラメーターに対しても使用できます。

```
{config.$directory}/html
```

7. 既に述べたことですが、他のユーザーによってデプロイおよび実行されるソリューションを作成する場合は特に、そのようなユーザーが IBM Security Directory Integrator のスキルを持っていることを期待してはなりません。テストおよび検証用の場合も含め、AssemblyLine を開始するためのバッチ・ファイルまたはスクリプトを用意してください。これらは、例えば、単にデータ・ソースに接続して、その接続の成否を報告するだけのものの場合もあります。バッチ・ファイルまたはスクリプトでステータス情報を返す目的でメッセージをコンソールのコマンド行に出力するには、チュートリアル練習で使用した AL バージョンの `task.logmsg()` ではなく、サーバー・メソッド `main.logmsg()` を使用することに注意してください。 AL バージョンの呼び出しは、メッセージをログに送るだけです。

以上のガイドラインはごく一部です。詳細については、他の IBM Security Directory Integrator 資料およびニュースグループを参照してください。

ロギングおよび監査

IBM Security Directory Integrator では、柔軟なログ管理を行うために *log4j* を使用しています。Unix syslog、Windows eventlog、日々のファイル、ローリング・ログなどを含むいくつかの標準的な *Appender* の中から選択できます。新規の Appender を作成するかダウンロードして使用することができます。

デフォルトでは、最小限のサーバー・ロギングが使用可能になります。最小限の作業として、FileRoller Appender を使用し、ソリューション・ディレクトリーのサブフォルダー「`logs`」に対して書き込みを行い、log-file を AssemblyLine と同じ名前をつけて、AssemblyLine 用のロギングを定義する必要があります。したがって、「CSV2XML」AL に対しては、ローリング・ログ・ファイルのセットを次のファイル・パスをベースに定義します。`logs/CSV2XML.log`

`logmsg()` メソッドでは、以下のキーワードをログ・メッセージの直前の最初の引数として渡すことにより、オプションでメッセージにログ・レベルを定義できます。DEBUG、INFO、WARN、ERROR、FATAL ログ・レベルは包括的なので、WARN は ERROR および FATAL を含み、DEBUG はすべてのレベルのメッセージがログに記録されることを意味します。例えば、

```
task.logmsg("DEBUG", "Updated: " + conn);
```

のようなメッセージを発行するのは、DEBUG レベルのロギングが設定された Appender だけです。

ソリューションに監査メッセージを追加して、実行中のサーバーの外部からメッセージのオンとオフを切り替えられるようにすることが可能です。オンとオフの切り替えは、`task.logmsg()` 呼び出しに対する呼び出しに、プロパティーの値を検査する IF 文の接頭部を付けることによって行います。例えば、次のようなスクリプトのスニペットが、「DataFlow - Update Successful」フックに現れます。

```
if (system.getTDIProperty("MyProps","audit").equalsIgnoreCase("true"))
     task.logmsg("DEBUG", "Updated the following data: " + conn);
```

tdisrvctl コマンド行ユーティリティーを使用して「MyProps」プロパティー・ストア内の「audit」プロパティーの値を変更することで、実行中のサーバーのこのタイプの監査メッセージのオン/オフを動的に切り替えることができます。

一般に、ログ情報が多過ぎる方が少な過ぎるよりもいいと言えます。しかし、一方でログ出力を必要以上に大量に出すべきではありません。乱雑なログ出力の中から必要なメッセージを探し出すのが困難になります。

接続の問題

接続の問題は通常、初期化エラーと接続の損失という二つのカテゴリーに分類できます。

デフォルトでは、その初期化フェーズの間の AL 処理がまさに開始される時点で、すべてのコンポーネントが接続を開始します。何らかの理由によりこの時点で接続に失敗した場合には、警告を発するか、さらにはパラメーター設定を変更して再接続を試みるスクリプト・コンテナーを提供する「プロローグ - 接続エラー発生」フックが起動されます。 同様に、AL サイクルの実行中に接続エラーが発生した場合には、この状況を処理する「データ・フロー - 接続損失」フックを使用します。

このカスタム処理に加えて、コネクターと関数コンポーネントも「**接続エラー**」タブを介して、組み込みの*再接続*機能を提供します。ここで、損失した接続を再度確立させるか、初期化問題の場合には接続セットアップの試行を続行させるよう、コンポーネントに指示することができます。

初期化エラーの場合には、SSL ネゴシエーション中の散発的なタイムアウトといったような繰り返される問題、あるいは類似の散発的な接続の問題を経験しているのでなければ、再接続させることはめったにありません。

しかし接続損失の際には、コンポーネントに接続を再確立させて何事もなかったかのように続行させる「**接続損失時の自動再接続**」が推奨されます。*イテレーター・モード*のコネクターの場合には、結果セットの最初の項目でサイクルが再開されるので、再接続を行えば反復「カーソル」がリセットされるおそれがあることにも注意してください。もちろん、変更検出コネクターのような状態検知イテレーターならば、コンポーネントが自動的に中断地点から再開する状態情報を使用するので、こうした問題は起こりません。

AssemblyLine の可用性

AL の可用性改善とは以下の 2 つのことを意味します。 1) AssemblyLine が停止することのないように対処すること、そして 2) 停止してしまった AL を可能な限り早く再始動させること。長時間にわたるマイグレーションや同期化などのシナリオにとって、障害が発生した時点で AssemblyLine が停止した場所から再び始動して続行することもまた必要です。

未処理の例外が発生すると、AssemblyLine は停止します。すべてのエラーを処理することで、この不測の事態に対する保護手段を取ることができますが、結局は少なくともIBM Security Directory Integrator 固有の個々の接続や関数コンポーネントの

「Default On Error」フックを作動させることになります。エラー・フックを作動させることで、例外が発生してもサーバーを稼働させ続けることができます。

チュートリアルの練習時に経験したように、エラーによって AssemblyLine が停止するときは、エラーの発生個所と原因についての情報を得るためにスタック・トレースを行います。エラー・フックを作動させることで AL を停止させずに済ませる場合は、スクリプトに用意されている専用のオブジェクトを使ってエラー・ステータスを報告する必要があります。

```
task.logmsg("ERROR", "[" + thisConnector.getName() + "] - " +
            error);
```

上記のスクリプトは 2 つのオブジェクトを使用した例です。実行中のスクリプトが関連付けられているコンポーネントを常に参照する `thisConnector` のオブジェクト、および `error` という名前の定義済み変数のオブジェクトが使われています。`work` や `conn` と同様に `error` オブジェクトは項目であり、「status」、「connectorname」[36] および「message」のような属性、加えて最新のエラー状態に関するその他の詳細の属性も有しています。これは項目オブジェクトなので、たとえば `work.message` のような属性名への直接参照と同様に `task.dumpEntry(error)` を使うか、あるいは、すべての項目オブジェクトは必要とされる文字列表現に変換することができるため、上例のように単に `error` を文字列メッセージに追加することで、内容を表示することができます。

スクリプト中で発生した例外を処理するために、属性マップ割り当てやフックおよびスクリプト・コンポーネント中でのように、コードを try catch ブロックでラップします。こうすると、例外をキャッチし、自身で処理できるようになります。

```
try {
 res = myLib.callToSomeFunctionThatMightFail();
} catch (excptn) {
 task.logmsg("Call failed with error: " + excptn);
}
```

エラー処理に加えて、前セクションに記されていた自動再接続機能を使用します。これによって、AssemblyLine がデータ・ソースやファイアウォールによってタイムアウトになるといったような、一時的な接続の問題から停止することを防ぐことができます。

それでも、何らかの理由で AssemblyLine が早々と停止してしまう場合、次に進む手順はそれを再起動させることです。アプローチの 1 つは障害/応答動作を定義する Web 管理ツールを使うことです。

もう 1 つの、相補的でもあるアプローチは、「ランチャー」AL を作成して、前のセクションのチュートリアルの練習で使用した AssemblyLine 関数コンポーネントを利用することです。この AL 関数コンポーネントを決して停止することのない、言い換えるならば常に *true* 条件である「ConditionalLoop」コンポーネントに配置し、必要なサービス AL を AL 関数コンポーネントが呼び出してそれが完了するまで待つように構成しておけば、制御が「ランチャー AL」に戻る時にはいつでも、無限ループによってデータ・フローが再開されるようにできます。

---

36. 「コネクター」という語は「コンポーネント」と同義語である場合があるので、 `thisConnector` のような変数が FC や SC、属性マップ・コンポーネントのことをも指すことがわかります。同様なことは、`error` オブジェクトの「connectorname」属性にも当てはまり、どのようなタイプのコンポーネント名も保持することができます。

上記の再開ループ技法を変更検出コネクター (またはデルタ・エンジン、どちらも他所に記述) の 1 つに基づいた同期 AL に適用すれば、AssemblyLine は停止した地点から自動的に続行されます。適用しないと、状態処理をユーザー自身で行う負荷が発生します。一般的な技法としては、タイム・スタンプやソート済み結果セットのための他のキー値といった状態情報を保持しておくためにシステム・ストアを使います。そうしておけば、AssemblyLine の初期化が行われる時にはいつでも、この状態情報がイテレーター・コネクターに適用され、以前に処理された項目の後に直ちに処理を再開できるようになります。

もし AL 用の反復データ・ソースがソート済みのリターンをサポートしていない場合には、反復を最初から再開させる必要がある場合もあります。この場合に、出力マッピング属性をターゲット・システムで検出された最新のものと比較して、差異がなければ修正処理を行わないという「**変更の計算**」機能が、「コネクター更新」モードによって提供されていることに注意してください。

IBM MQ のような複数 IBM Security Directory Integrator サーバー間の機密保護機能のあるトランスポートを導入することで、単一障害点を回避することができます。このようにして、キューの中にデータや処理命令をも配置することで、多くのサーバー (そして AssemblyLine) を処理の開始に使用できるのです。受信する側の多重サーバー/AL は先着順にこれらを抽出して、要求された処理を実行します。これはより堅固なソリューションとなるばかりではなく、送信側と受信側の AssemblyLine を追加することでスケーリングが容易になります。

本書では、非常に簡単ですが、かなり大きな主題についての解説を行いました。 むしろ、特定のアプローチという処方箋よりもひらめきの方が決め手になります。より具体的な推奨や実例については、コミュニティー Web サイトやディスカッション・グループに当たってみることをお奨めします。

スケーリングとパフォーマンス

このトピックについても、本書で十分に解説することはできません。しかし、いくつかのポイントについて、概略を示します。

データ・フローの速度を低下させる主な要素が I/O です。ソースからのデータの取り出しや、ターゲットへ押し込むデータの変更に要する時間は、AL 内部の処理時間に非常に影響します。さらに、接続のネゴシエーションも、特にセキュリティー・ハンドシェークが絡んでいるときは、非常に時間がかかります。これらのことを踏まえて AssemblyLine を設計して作成することは、性能に直接効果を与えます。

例えば、前回の練習で作成した HTTP Server ソリューションのように、クライアントからの着信プロトコル要求を受信するサーバー・モード・ベースの AL について考えてみましょう。受信された要求はそれぞれ、サービス AssemblyLine を起動して実際の処理を実行するようにします。この呼び出された AL でコンポーネントを初期化する必要がある場合には、それぞれの要求のターンアラウンド時間は、最短でもすべての接続時間を合計した長さになります。

この状態を改善する方法は 3 つあります。

- 実際の処理をディスパッチする代わりに、メインのサーバー・モード AL で行うようにする。これにより、要求ごとに接続をセットアップして切断する必要がなくなります。
- サービス AssemblyLine が *Manual/Cycle Mode* で呼び出されるように設計する。Manual/Cycle モードでは、サービス AL は AL FC の初期設定時に初期化を行います。さらに、AssemblyLine 関数コンポーネントによって、呼び出しごとにサービス AL が 1 サイクルだけ繰り返されます。この方法は、サービス AL が呼び出し側の AssemblyLine のコンポーネントのように動作するので、この動作に適合するようにサービス AL を作成する必要があります。
- グローバル・コネクター・プーリングを利用する。この機能は、「リファレンス」で説明されているように、サーバーの起動時に初期化され、必要に応じて、複数の AssemblyLine の間で共有されるコネクターのプールを定義できるようにします。

性能を向上させるためのもう 1 つの方法は、同時に存在する複数の AssemblyLine にまたがる膨大な処理タスクを分割することです。例えば、1 つの AssemblyLine でソースのデータ・セット全体を処理の対象とする代わりに、複数の AL を起動して各々がサブセットを処理する、マイグレーション・タスクを考えます。AssemblyLine を呼び出すときに初期化のパラメーターを渡すことができるので、インスタンスが処理するデータ範囲を制御するフィルター・パラメーターと共に複数回開始される単一の AL を開発できます。

別の手法として、前のセクションで説明したように、メッセージのバスをソリューションに取り込む方法があります。この方法は、IBM の大規模クライアントに採用されて大成功を収めています。

ネットワークのリンクが不安定なために、あるいはシステムの可用性が低いために、処理が妨げられて速度が低下している場合、バックグラウンド・タスクの働きをする AL を追加して、入手するのが困難なデータをローカルの高速な保管場所に同期化させるよう展開することもできます。この方法は、特に、リアルタイムのサービスをインプリメントするときに、クライアントの要求への十分な応答時間を確実に得るのに役立ちます。

監視

IBM Security Directory Integrator ソリューションは最初から、Web 管理ツールを使用して、素早く簡便に管理できます。この統合サービス・コンソール (ISC) プラグインは AppServer アプリケーションで、インフラストラクチャー内の任意の数のサーバーで実行されている任意の数のソリューションをモニターできます。AL 統計、ログ、およびスタート・ストップ時刻の表示という最初から使用できる機能に加えて、Web 管理ツールでは、ヘルスおよびインシデントのコンソールをカスタマイズし、さらにスケジュールおよび障害/対応動作を定義して、AssemblyLine の正常な動作を維持します。

また、IBM Security Directory Integrator ソリューションを構成して、状況イベントを (例えば、SNMP トラップとして、またはシステムのカスタム・イベント・フォーマットを使用して) 送信することもできます。システムは、ITM などを使用した JMX 管理および監視用に、簡単に構成できます。

AssemblyLine デバッガー

繰り返しますが、これまで述べたように、時間をかけて AssemblyLine Debugger に熟達しておくと、ソリューションの開発、トラブルシューティング、および導入に要する時間が 10 分の 1 に短縮されて報われるようになります。

付録. EasyETL ガイド

「ETL」とは、抽出 (Extract)、変換 (Transform)、およびロード (Load) の略であり、データをある場所から取得し、必要に応じて変換してから、他の場所に置く、ということを要約した用語です。「EasyETL」は IBM Security Directory Integrator の機能の 1 つであり、この ETL を、少ないキー・ストロークで、素早く、対話式に実行できるようにしたものです。

一般的な ETL の例

- データベース・レコード、IBM Notes 文書、ディレクトリー項目、または受信メールや MQ メッセージなどのファイルへのエクスポート。
- ファイルからシステム、またはデータ・ストアへのデータのロード。
- ソフトウェア/スキーマの更新の場合の、システムから他のシステムへのデータの直接移行、または同じシステムのバージョン間でのデータの直接移行。

IBM Security Directory Integrator EasyETL を使用すると、このような例やその他のシナリオを、直観的に分かる少ないステップで解決することができます。使用するインフラストラクチャーのミッション・クリティカルなデータ・フローだけでなく、1 回のみのデータ移動のニーズにも適したソリューションを提供します。

新規の EasyETL タスクを作成する最初のステップは、使用する入力ソースを選択し、転送する属性を選択することです。この時点で既に、IBM Security Directory Integrator では EasyETL ジョブを実行し、貼り付けに使用するコピー・バッファーに読み取るデータを収集することができます。データを変換または計算する必要がある場合は、EasyETL で、変換の追加、ETL ジョブの再実行、および変換されたデータのコピー・アンド・ペーストを行えます。また、出力先を選択して、EasyETL ジョブにそこにデータを直接書き込ませることもできます。

EasyETL ソリューションが要求どおりに動作したら、統合タスクを起動およびスケジューリングするためのコマンド行資産 (バッチ・ファイルまたはスクリプト) を IBM Security Directory Integrator で生成できます。最後に、EasyETL は IBM Security Directory Integrator の変更検出機能を活用し、ETL ジョブをデータ同期タスクに素早く変えます。

注: 各 EasyETL ソリューションが、単一の AssemblyLine の IBM Security Directory Integrator プロジェクトであり、フル機能の開発環境でオープンできることは、IBM Security Directory Integrator ユーザーにとって朗報です。ただし、そこで変更してしまうと、 EasyETL としてはもう使用できなくなります。

EasyETL の使用

IBM Security Directory Integrator 構成エディターを起動し、使用するワークスペースを選択します。IBM Security Directory Integrator を初めて開始した時には、ウェ

ルカム・ページが表示されます[37]。

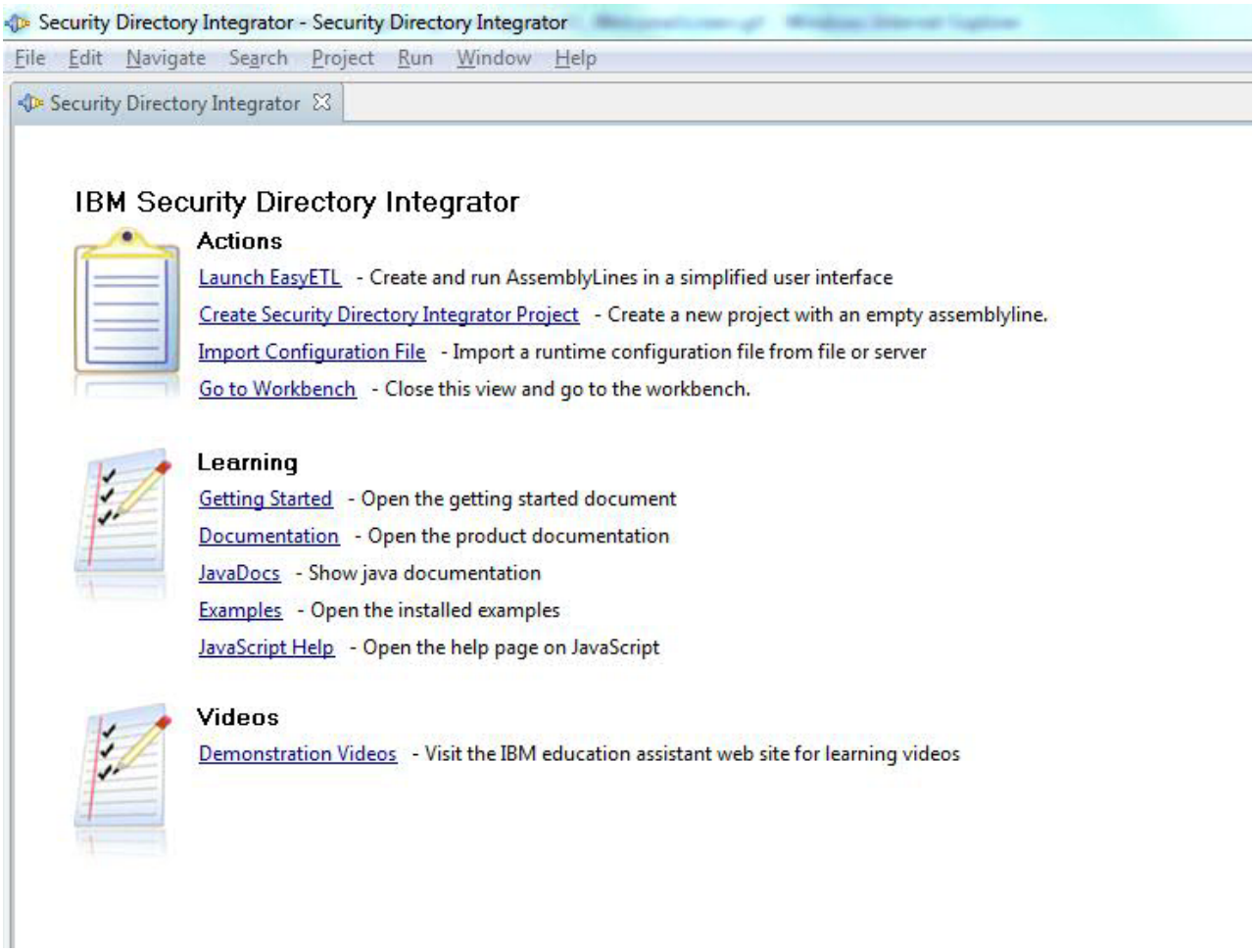

図 87. ウェルカム画面

上のスクリーン・ショットに表示されているように、ここの一番上のリンクを選択すると、EasyETL[38] ワークベンチが開きます。ここで、このリンクをクリックして、Security Directory Integrator の EasyETL ワークベンチを開きます。

37. ウェルカム・ページには、メインメニューの「**ヘルプ**」 > 「**ようこそ**」を選択することで、いつでも戻ることができます。

38. EasyETL は、Security Directory Integrator のパースペクティブ です。メニューで「**Windows**」 > 「**Open Perspective**」 > 「**Other...**」を選択すると、パースペクティブ間で切り替えることができます。パースペクティブに変更を加えてから、リセットしてデフォルトに戻す場合は、「**Windows**」 > 「**Reset Perspective**」を選択するだけで可能です。

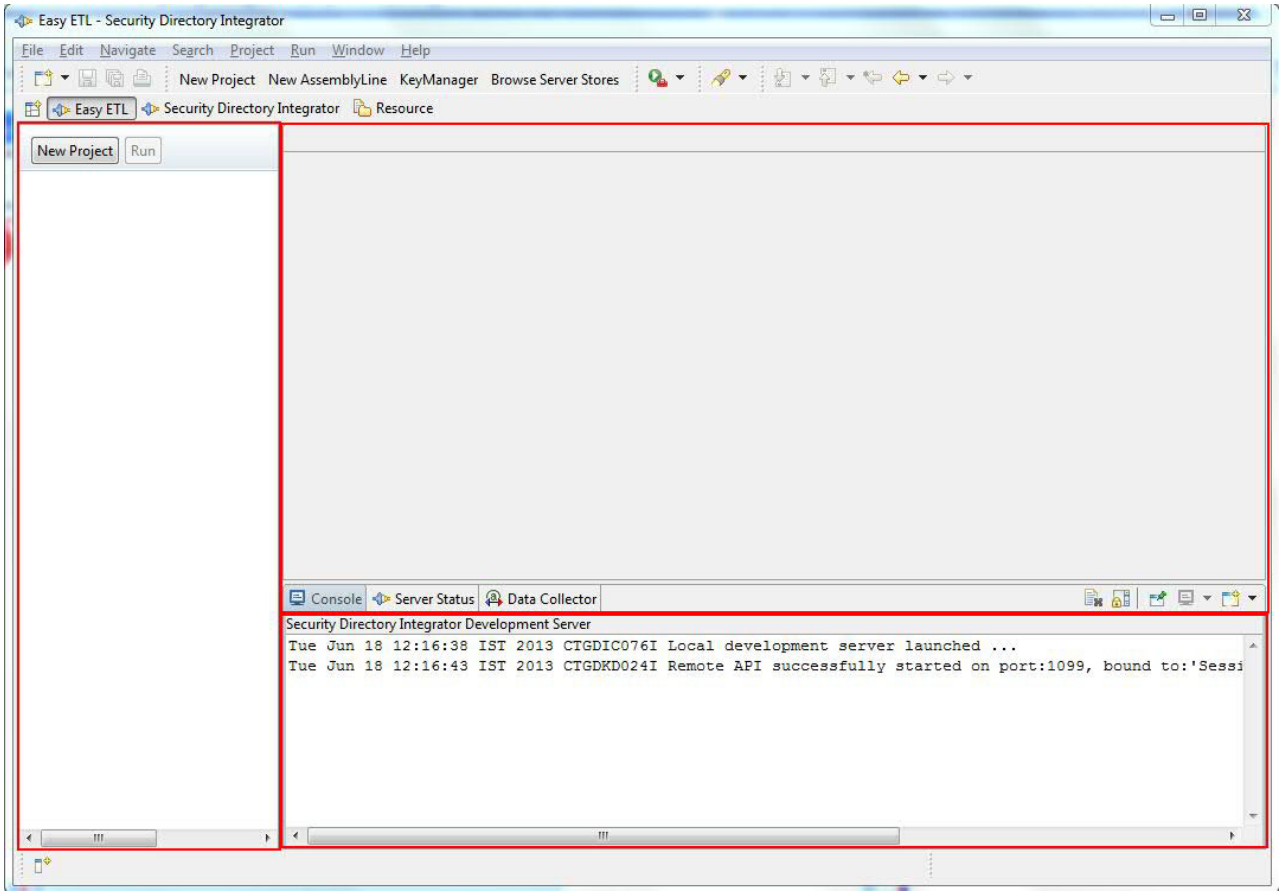

図 88. EasyETL ワークベンチ

EasyETL ワークベンチには、次の 3 つが表示されます。

- ETL ジョブをリストするプロジェクト・ナビゲーター。操作したいプロジェクトを右クリックすると、そのプロジェクトの実行や、そのプロジェクトを起動するためのコマンド行資産の作成などを行うことができます。
- オープンされている各プロジェクトの単純な AssemblyLine[39] エディター。
- 画面の下部に並ぶタブで示されるさまざまなビュー。デフォルトでは 3 つのビューがあります。
  - テスト IBM Security Directory Integrator サーバーからのコンソール出力。
  - サーバーおよび実行中の EasyETL プロジェクトの両方をモニターできる、サーバーの状況ビュー。
  - 各サイクルの結果データが、表形式でリストされるデータ・コレクター。

この文書を読み進めていくと、詳細が各セクションで説明されています。

39. 「AssemblyLine」は IBM Security Directory Integrator のデータ・フローのインプリメンテーションです。そのため、EasyETL プロジェクトを作成、またはオープンすると、基礎となる AssemblyLine がエディターに表示されます。

「AssemblyLine」という用語は、この文書および他の IBM Security Directory Integrator 文書で「AL」に省略されますのでご注意ください。

プロジェクトの作成

プロジェクト・ナビゲーターの上部にある「**新規プロジェクト**」ボタンを押して、新規の EasyETL プロジェクトを作成します。

図 89. 「新規プロジェクト」ボタン

このプロジェクトに「CSVtoXML」という名前を付けて、「**完了**」を押します。これで、単純な AL エディターに、作成した新規のプロジェクトがオープンします。

図 90. 単純な AssemblyLine エディター

エディターには 2 つのドロップダウンがあります。一方は入力ソースを選択するもので、もう一方はターゲットを選択するものです。下の領域には、ソースを選択するまで何も表示されません (支援テキストは除く)。

ETL AL の入力のセットアップ

ソース情報のドロップダウンをクリックして「**ファイル・コネクター**」を選択し、EasyETL AssemblyLine の入力を構成します。

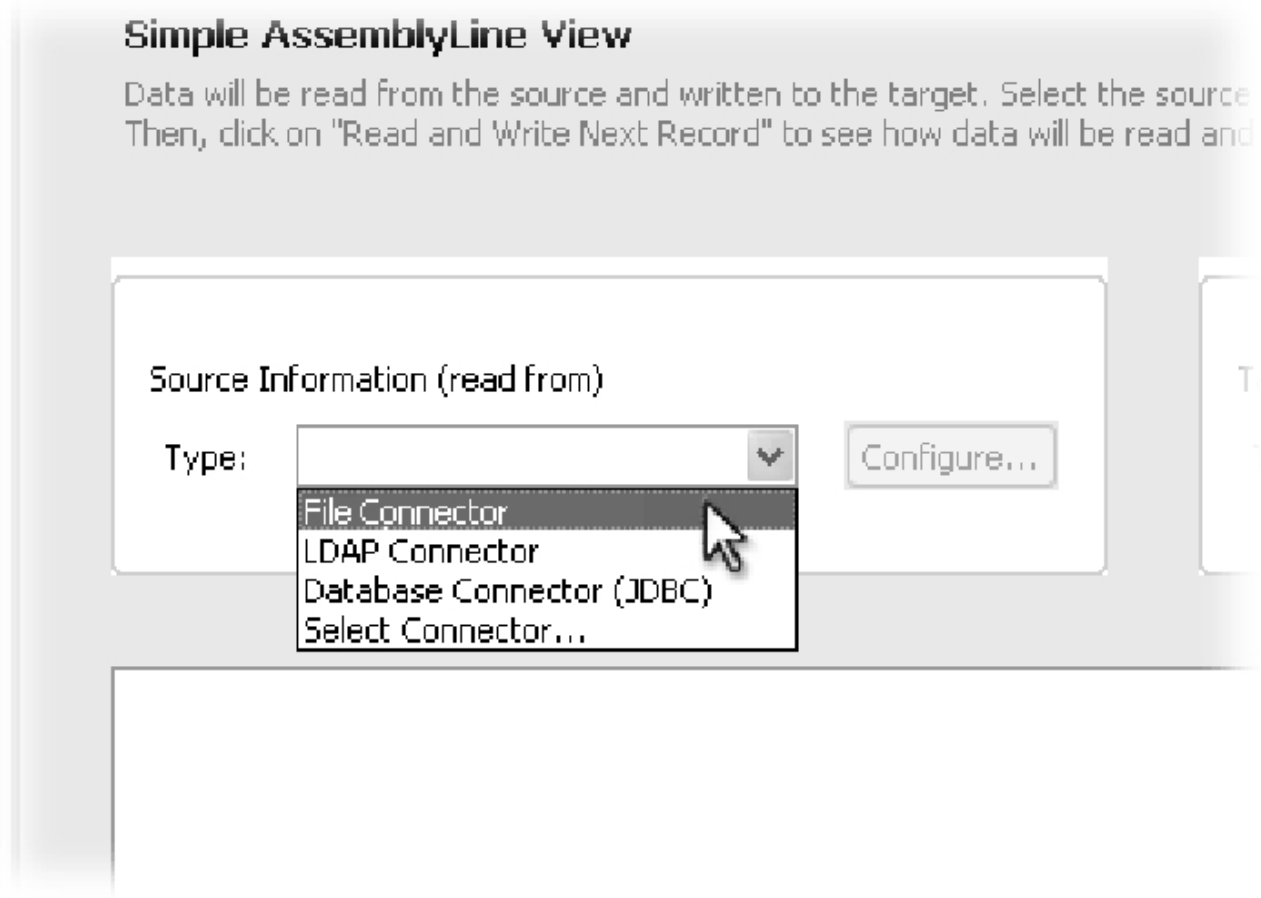

図 91. ソース情報の選択

選択したコネクターの構成ダイアログが表示されます。

「ファイル・パス」パラメーターに、以下にある「`People.csv`」ファイルを指定します。

TDI_HOME`/examples/Tutorial`

ここで、*TDI_HOME* は、ご使用のマシンの IBM Security Directory Integrator インストール・ディレクトリーで置き換えます[40]。

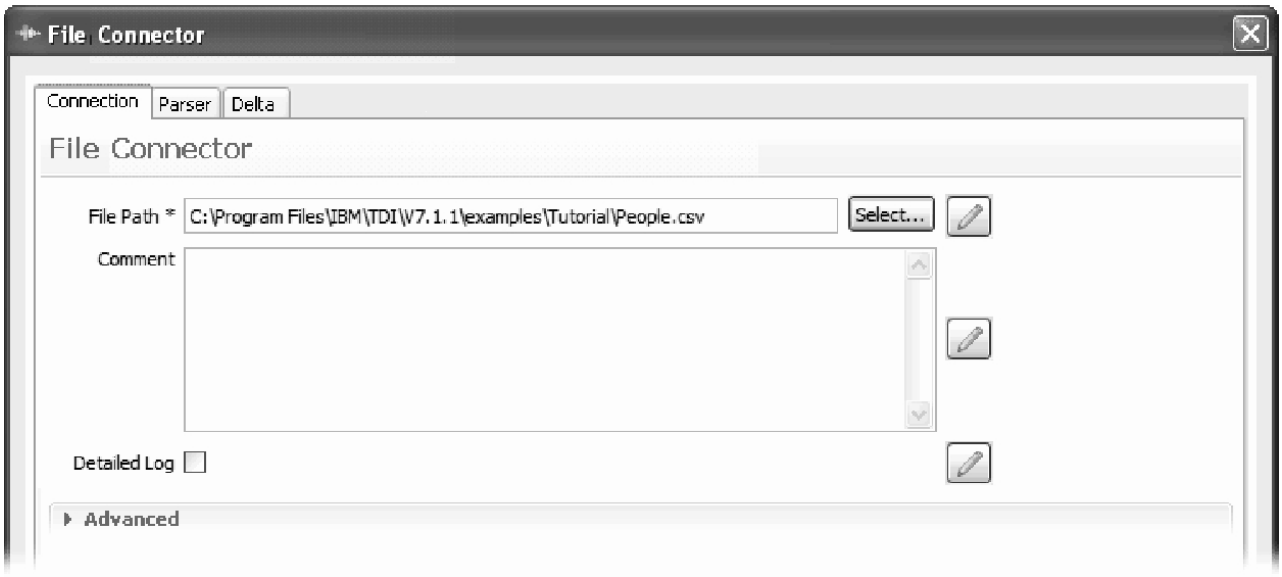

図 92. ファイル・パス・パラメーターの設定

ここで、「**パーサー**」というラベルのついたタブをクリックし、「**CSV パーサー**」を選択します。このとき、デフォルトの構成パラメーターはそのままにしておきます。最後に、ダイアログ・ボックスの下部にある「**接続**」ボタンを押して、接続をテストし、使用可能な属性を検出します。

40. ここで留意することは、IBM Security Directory Integrator を、 Windows 上で実行している場合、パス分離文字として、スラッシュおよび円記号 (¥) の両方がサポートされることです。スラッシュを使用すると、IBM Security Directory Integrator を実行する Windows とその他のすべてのプラットフォームの間で、IBM Security Directory Integrator ソリューションをより移植しやすくなります。

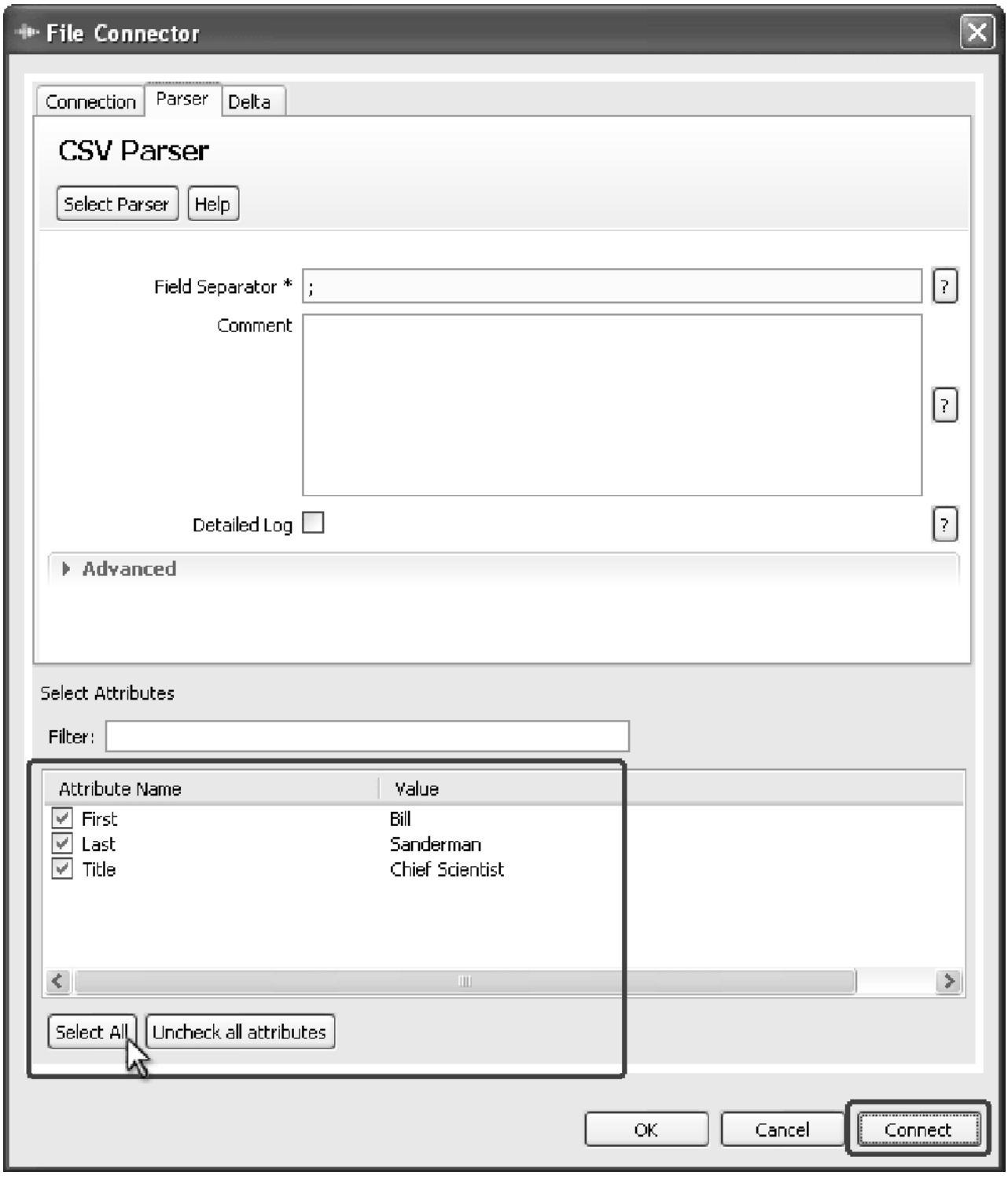

図 93. 接続テストおよびスキーマの検出

接続したシステムのスキーマが表示されるので、データ・フローで使用するものをここから選択することができます。この例では、「**すべて選択**」ボタンを使用してから「**OK**」を押して構成ダイアログを閉じます。

EasyETL ワークベンチに戻ります。今構成した入力ソースが反映されて、単純な AL エディターの下半分が変更されています。このとき、IBM Security Directory

Integrator は、この情報を一度に 1 レコードずつステップスルーするボタン、ETL タスクを完了するまで実行するボタン、および停止するボタンを提供します。

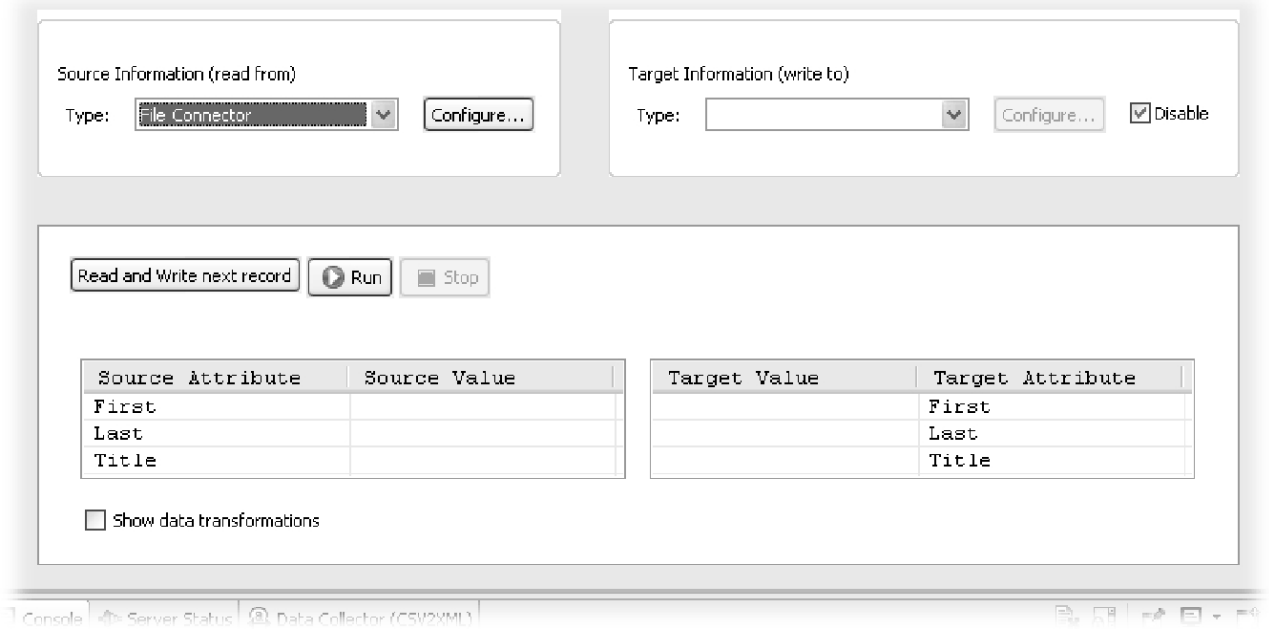

図 94. 構成された入力ソース

これらのボタンの下には、2 つのグリッド・ボックスが表示されます。これらをデータ・ビューアーと呼びます。ここには、データ・フローで扱われる属性がリストされます。左側のデータ・ビューアーには、入力属性が表示されます。前のステップで、読み取り用に選択した属性が、上部に太字で表示されます。これを入力マップ と呼びます。下部には、選択されなかった属性がグレイで表示されます。これらの属性をダブルクリックして、入力マップに組み込むことができます。同様に、マップから属性を除去するには、属性をダブルクリック、または削除します[41]。

右のボックスは出力ビューアーで、書き込まれる一連の属性が表示されます。IBM Security Directory Integrator 用語で、これを*出力*マップ と言います。デフォルトでは、入力マップ用に選択したリストと同一です。出力属性の名前は、右側の列をクリックして値を編集すると、変更することができます。この方法を用いて、「Title」という属性の名前を「JobTitle」に変更します。

41. 属性を右クリックして、「**属性の追加**」または「**属性の除去**」を選択することもできます。

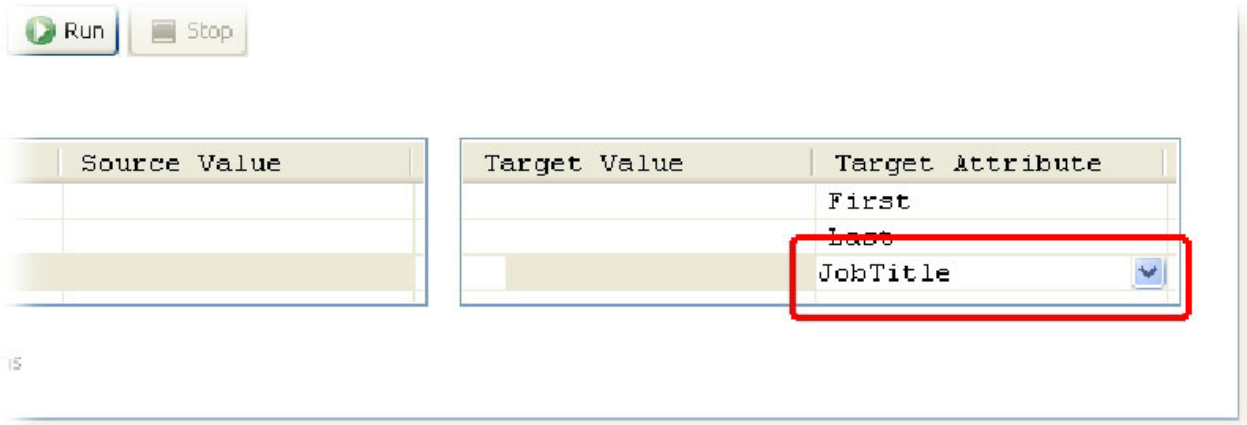

図 95. 出力属性の名前変更

これで、作成した EasyETL プロジェクトは、最初のテストの準備ができました。

EasyETL AssemblyLine の実行

画面下部で「**データ・コレクター**」ビューを選択し、「**次のレコードの読み取りおよび書き込み**」ボタンを押します。これにより次のことが起こります。

1. EasyETL AssemblyLine が実行中のサーバーに転送され、開始されるので、遅延が発生します。
2. 最初のレコードが CSV 入力ソースから読み取られ、解析され、データが入力および出力グリッドに表示されます。
3. 出力用に選択した属性は、「データ・コレクター」ビューに書き込まれ、収集されます[42].

出力ターゲットをまだ選択していない場合でも、データが AssemblyLine を流れている様子を表示しながら ETL プロジェクトを実行およびテストすることができます。

42. 「**データ・コレクター**」ビューの上部の一番左のボタンを押すと、構成ダイアログが開きます。ここで、「**データ・コレクター・バッファー**」のサイズを増やすことができます。 大容量のデータを収集する計画がある場合は、IBM Security Directory Integrator で使用可能なメモリーを増やす必要があることにご注意ください。これを行うには、IBM Security Directory Integrator のインストール・フォルダーで「`ibmditk`」バッチ・ファイル/スクリプトを見つけ、それをエディターで開きます。下部の方に「`miadmin`」を起動する行があるので、`-vmargs` オプションの後に、テキスト `-Xmx512M` を挿入します。この例では、IBM Security Directory Integrator データ・メモリーは 512 M バイトに増えます。

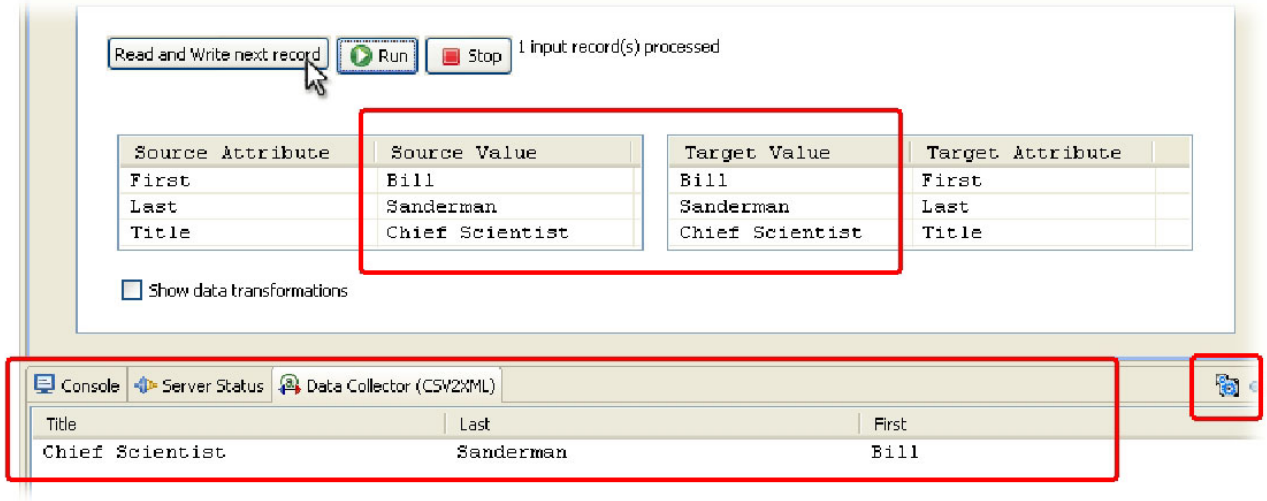

図 96. 1 件のレコードが読み取られて収集された

「**次のレコードの読み取りおよび書き込み**」ボタンを押すたびに、次のレコードが読み取られ、表示および収集されます。ここで「実行します」ボタンを押すと、ETL ジョブが完了するまで実行されて、次の AL レポートが表示されます。

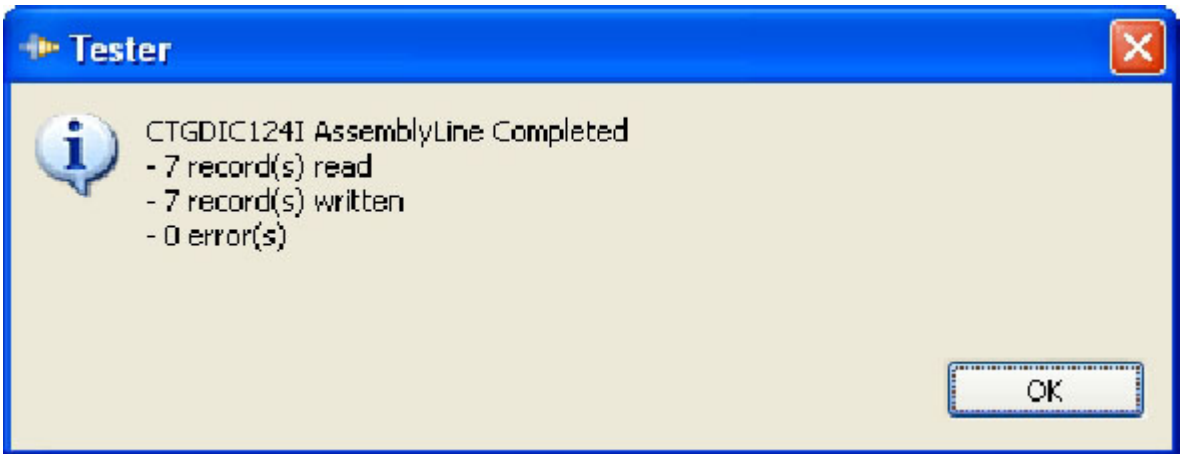

図 97. 完了した EasyETL AssemblyLine

上記のダイアログに示されるように、実際に書き込まれたレコードはありません。しかし、「**データ・コレクター**」は、抽出および転送された情報の有益なフィードバックを表示します。

さらに、収集されたデータの行を選択し、この情報をファイル、または他のターゲットにコピー・アンド・ペーストすることができます[43]。

43. データは、スプレッドシートやレポート・テーブルに値を簡単にインポートできるよう、CSV フォーマットでコピーされます。「CSV」は文字区切り値 (Character Separated Value) を示し、 IBM Security Directory Integrator で使用される分離文字はセミコロン (;) です。

変換

ここまでは、出力値は入力と同じ値でした。しかし、読み取ったデータを基に、出力値を操作または計算したい場合もあるでしょう。これには、IBM Security Directory Integrator EasyETL に JavaScript で変換を書き込みます。

変換を行うには、まず最初に「**データ変換の表示**」チェック・ボックスを選択して、変換を使用可能にする必要があります。

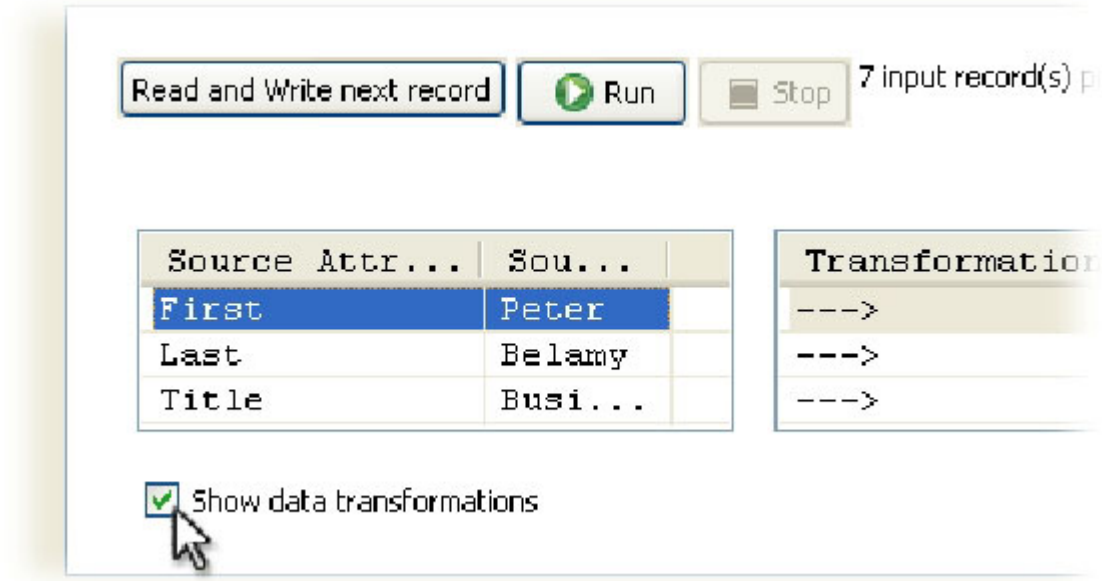

図 98. 変換の有効化

これにより、2 つのデータ・ビューアーの間に、新しいグリッド・ボックスである変換ビューアーが表示されます。ここには、3 つすべての出力属性が、入力属性から直接値を取得していることを示す矢印が表示されています。言い換えれば、変換は行われていません。これから新規の出力属性を定義し、変換スクリプトを追加してその値を計算します。これには、出力マップで右クリックし、「属性の追加」を選択して名前を「FullName」にします。この新規の出力属性の左側の「変換」をダブルクリックし、次のスクリプト・スニペットを入力します。[44]

```
return First + " " + Last
```

---

44. Ctrl + スペース・キーを押すと、入力できる内容の提案リストを表示できます。このリストには、「タスク」や「メイン」などの特別なオブジェクトも含まれ、下部には、入力ソースから読み取られた属性がリストされます。

    また、ここに表示される、入力属性にアクセスするための表記は、EasyETL 内でのみ有効であることにご注意ください。フル IBM Security Directory Integrator ワークベンチでは、属性名を所有する項目オブジェクト (例えば、作業項目 (work)) を接頭部として属性名に付けなければなりません。結果として、上の例は次のようになります。

    ```
    return work.First + " " + work.Last
    ```

    作業項目および他の IBM Security Directory Integrator 概念については、この文書の最初のセクションで説明されています。

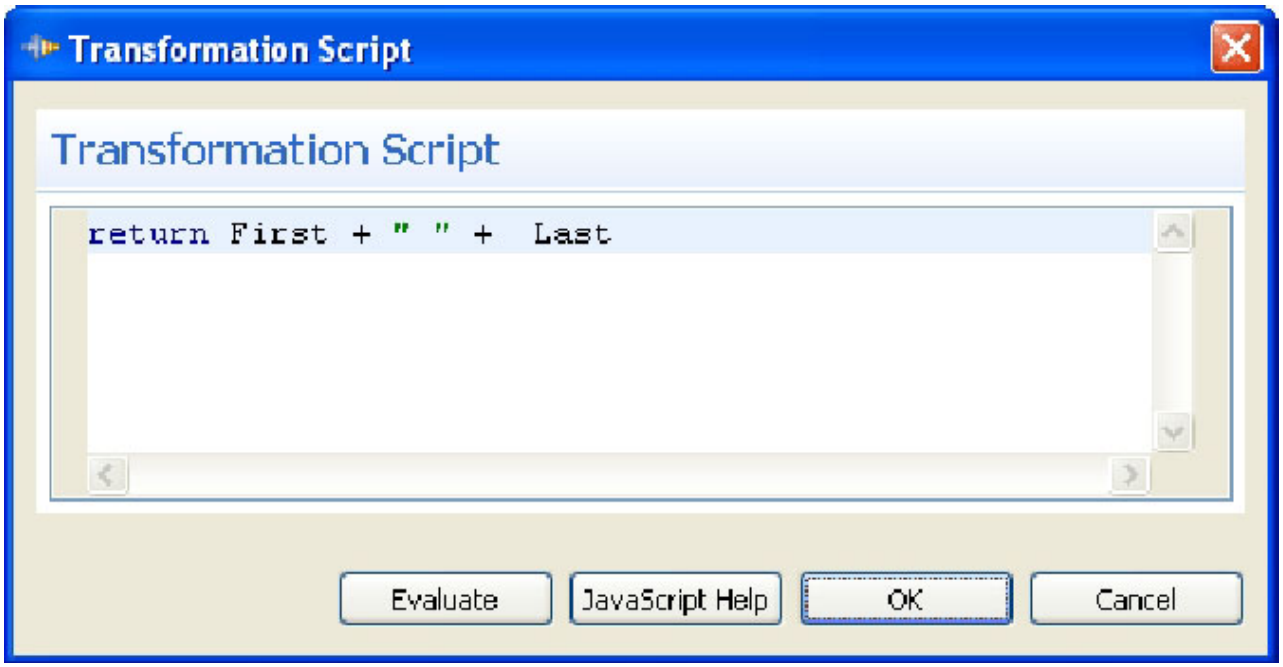

図 99. 変換スクリプトの表示

変換スクリプトのエディター・ダイアログには、JavaScript のヒントおよびサンプルを表示するボタンと共に、スクリプトをテストするための「**評価します**」ボタンがあります。

「**評価します**」ボタンを押して、どのように変換作業が行われるのか見てみましょう。

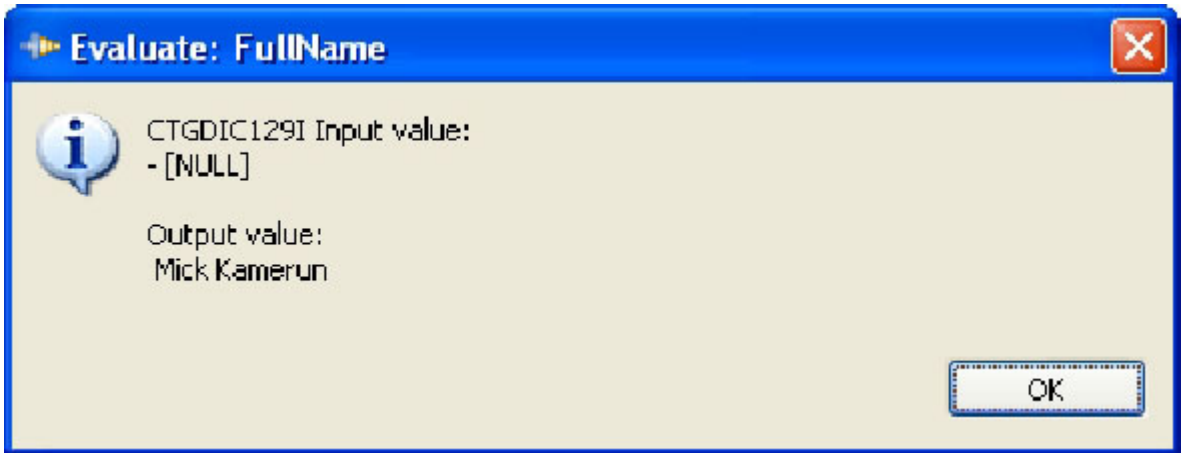

図 100. 評価式

表示される出力値は、AssemblyLine を実行した際に収集したデータを使用して計算されたものです。「**OK**」を押して「評価結果」ダイアログをクローズします。

「**OK**」を押してこのスクリプトを承諾し、変換スクリプト・エディターをクローズします。次に「**実行**」を押して EasyETL AssemblyLine を再実行してから収集された項目を表示します。**データ・コレクター**が、選択元となる 2 つのコンポーネン

ト・コレクション (「*出力*」および「入力」) をどのように表示するかに注目してください。「**出力**」を選択すると、作成した変換スクリプトが、各項目に対して、「FullName」の値をどのように生成したのかが分かります。

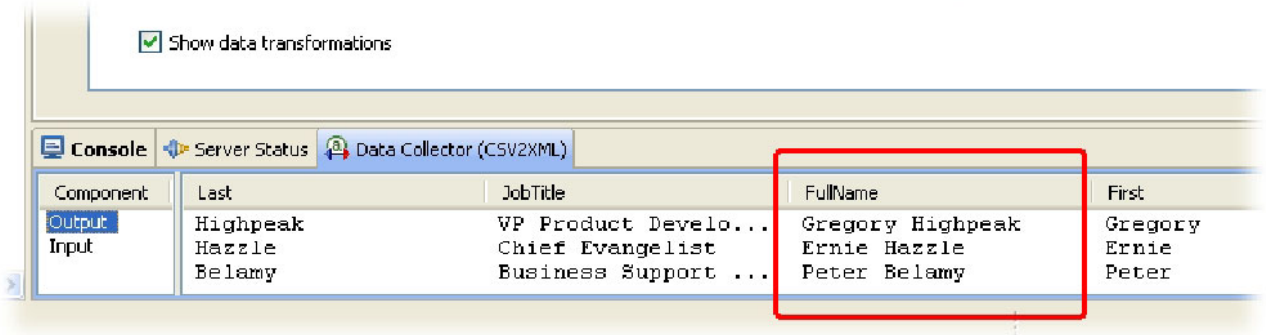

図 101. 計算された FullName 属性を持つ出力コレクション

これで、入力ソースのセットアップ方法、抽出する属性の選択方法、さらにこのデータを出力ニーズに合わせて変換する方法が分かりました。次のステップは、出力ターゲットを選択して、そこへデータを移動することです。

出力ターゲットの選択

出力ビューの上のドロップダウンを使用して、出力ターゲットの「**ファイル・コネクター**」を選択し、「Output.xml」というファイルへの書き込みをコネクターが行うようにします[45]。「**XML パーサー**」を選択して「**OK**」を押します。「接続します」ボタンを押して、入力したファイル・パスが有効であることを確認することもできます。しかし、コネクターに既存の XML ファイルを指定しなければ、検出されるデータはないでしょう。

出力が構成されたら、もう一度 ETL AssemblyLine を実行します。完了したら、出力ファイルをオープンし、結果を確認できます。

45. 絶対パスも相対パスも入力しない場合、IBM Security Directory Integrator は、インストール時に指定されたソリューション・ディレクトリーのパスをベースにします。

```
- <DocRoot>
  - <Entry>
      <First>Bill</First>
      <Last>Sanderman</Last>
      <Title>Chief Scientist</Title>
    </Entry>
  - <Entry>
      <First>Mick</First>
      <Last>Kamerun</Last>
      <Title>CEO</Title>
    </Entry>
  - <Entry>
      <First>Jill</First>
      <Last>Vox</Last>
      <Title>CTO</Title>
    </Entry>
  - <Entry>
      <First>Roger</First>
    </Entry>
  - <Entry>
      <First>Gregory</First>
      <Last>Highpeak</Last>
      <Title>VP Product Development</Title>
    </Entry>
  - <Entry>
      <First>Ernie</First>
      <Last>Hazzle</Last>
      <Title>Chief Evangelist</Title>
    </Entry>
  - <Entry>
      <First>Peter</First>
      <Last>Belamy</Last>
      <Title>Business Support Manager</Title>
    </Entry>
</DocRoot>
```

図 102. XML 出力

データベース・テーブルを出力ターゲットにしたり、IBM Notes アプリケーションや LDAP ディレクトリーから入力を行ったりすることも、簡単に行えます。この例

の ETL ジョブを作成するために行った手順は、作業を行うシステムまたはデータ・ストアに関係なく同じです。

変更の検出

IBM Security Directory Integrator は、入力データの変更を検出するためのいくつかの機能を提供します。変更検出コネクターのセットを提供するだけでなく、入力ソースのためのデルタ・エンジン を使用可能にするオプションもあります。

デルタ・エンジンは、データが読み込まれたときにスナップショットを作成し、それと、前の実行時に作成したスナップショットを比較して、何が変更されたのかを決定します。変更されていない項目はスキップされ、変更された項目だけが取り出されて EasyETL AssemblyLine で処理されます。

入力ソースの「**構成**」ボタンを押して、「**デルタ**」タブを選択します。

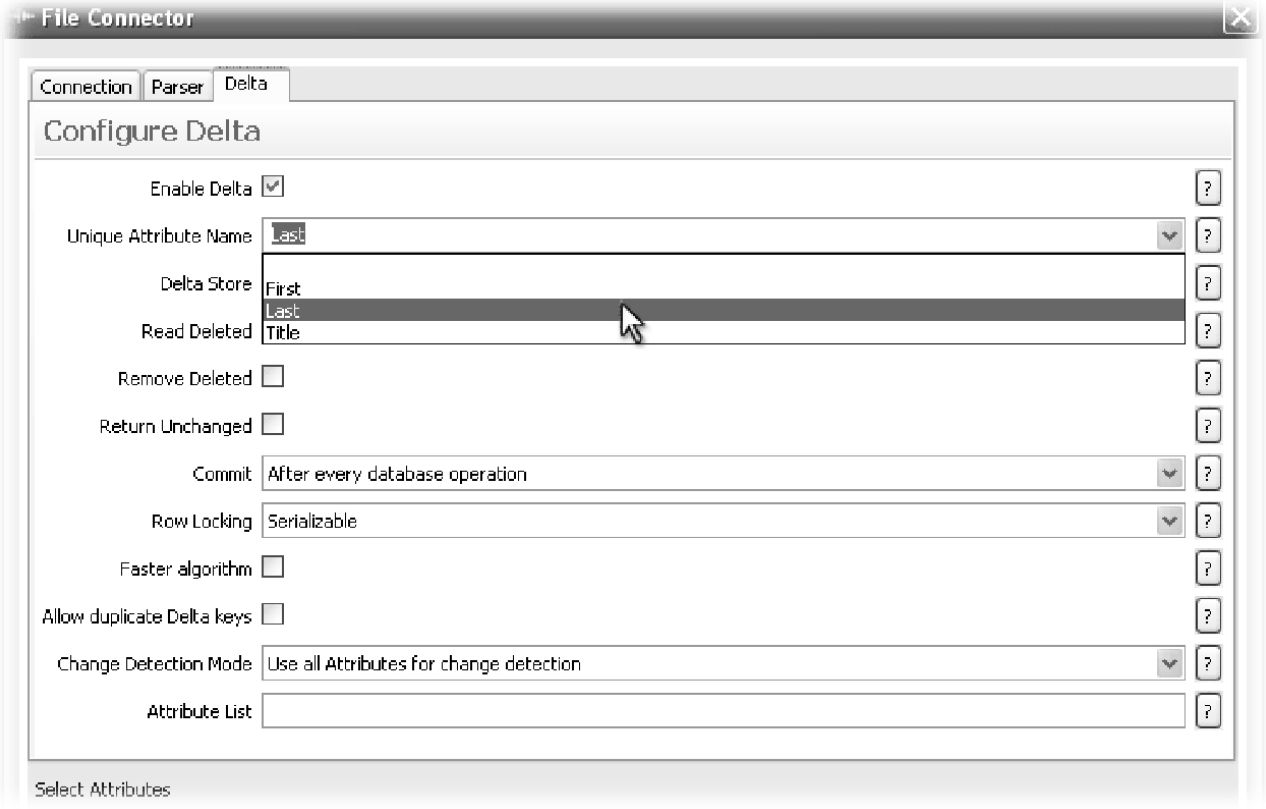

図 103. デルタの構成

まず、構成パネルの上部にあるチェック・ボックスを選択することで、デルタ・エンジンを使用可能にする必要があります。次に、ドロップダウンを使用して、「**固有属性名**」に「First」を選択します[46]。

46. ご推察の通り、デルタ・エンジンは、一意的にスナップショットを識別するために、入力属性の 1 つを使用します。入力データ内に使用可能な固有値がない場合は、複数の属性を指定することができます。これらの属性は連結されて、スナップショット ID を形成します。これを行うには、複数の属性の名前を正符号 (+) で分離して入力します。例: `First + Last`

ここで使用可能なパラメーターは他にもあります。そのうちのいくつかは、EasyETL ではなく、標準の IBM Security Directory Integrator ワークベンチで作業する場合に、より有効です。例えば、EasyETL AL は、新規の項目や変更された項目を検出および転送することができますが、データベースからの行の削除や、ディレクトリーの項目の削除は行いません。ただし、この情報を、LDIF パーサー付きのファイル・コネクターのような出力ターゲットに書き込みます。LDIF ファイルには、変更操作タグを含めることができます。LDIF インポートをサポートするシステムも複数あります。

IBM Security Directory Integrator の完全なデルタ処理機能についての詳細は、以下を参照してください。

http://www.tdi-users.org/twiki/pub/Integrator/HowTo/HowTo_SyncData_6.1.1070523.pdf

「**コミット**」パラメーターに変更を加えるとよいでしょう。このパラメーターは、新規および変更されたスナップショットをいつ IBM Security Directory Integrator システム・ストア・データベースにコミットするかを制御します。デフォルトでは「各データベース操作後」に設定されているので、読み取りフェーズで行われます。

ただし、スナップショットのコミット前に、変更が正常に転送されたことを確認したい場合は、このドロップダウンを「AL サイクルの終わり」に設定します。これにより、コミットは出力ターゲットが更新された後に行われます。

デルタ・エンジンが動作するためには、ベースラインとなるスナップショット・セットが必要です。これは、デルタを使用可能にした後、最初に ETL ジョブを実行すると作成されます。これが完了すると、ポップアップの書き込み発生のレポートが 2 倍になることに気付くでしょう。これは、IBM Security Directory Integrator がシステム・ストアに書き込まれるスナップショットも数えるため、処理される項目ごとに、2 回の書き込みが発生するからです。

EasyETL AssemblyLine をもう一度実行してみてください。今度は書き込まれる項目がないことが分かります。デルタ・エンジンは、入力レコードがすべて未変更であることを検出し、スキップしました。

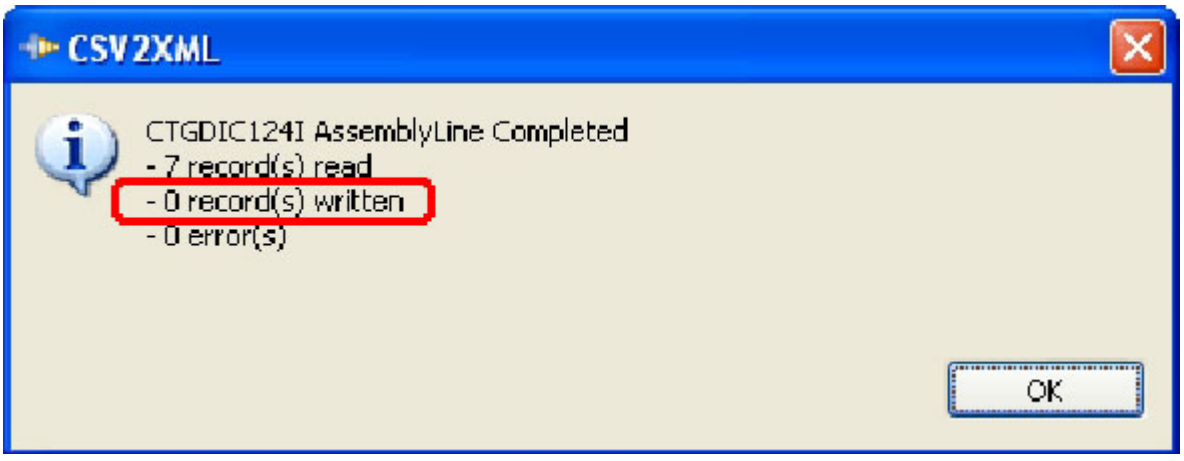

図 104. 未変更でスキップされたすべての項目

最終テストとして、入力 CSV ファイルを取り出し、「Last」以外のいくつかのフィールド値を変更します[47]。変更を保存して ETL ジョブを再実行します。変更した項目のみが処理されることが分かります。

更新用の出力ターゲットの構成

ファイルへの出力を行うための現在のセットアップはうまく動作しています。しかし、これらの変更を、ディレクトリー、RDBMS、または同様のデータ・ストアに出力していた場合は、既存のレコードを更新するだけでなく、新規データを追加したいでしょう。 EasyETL ジョブでこれを行うには、まず、変更するレコードの場所を特定する基準として使用する出力属性を選択する必要があります。

これには、目的の出力属性を右クリックし、「**リンク基準として使用**」オプションを選択します。

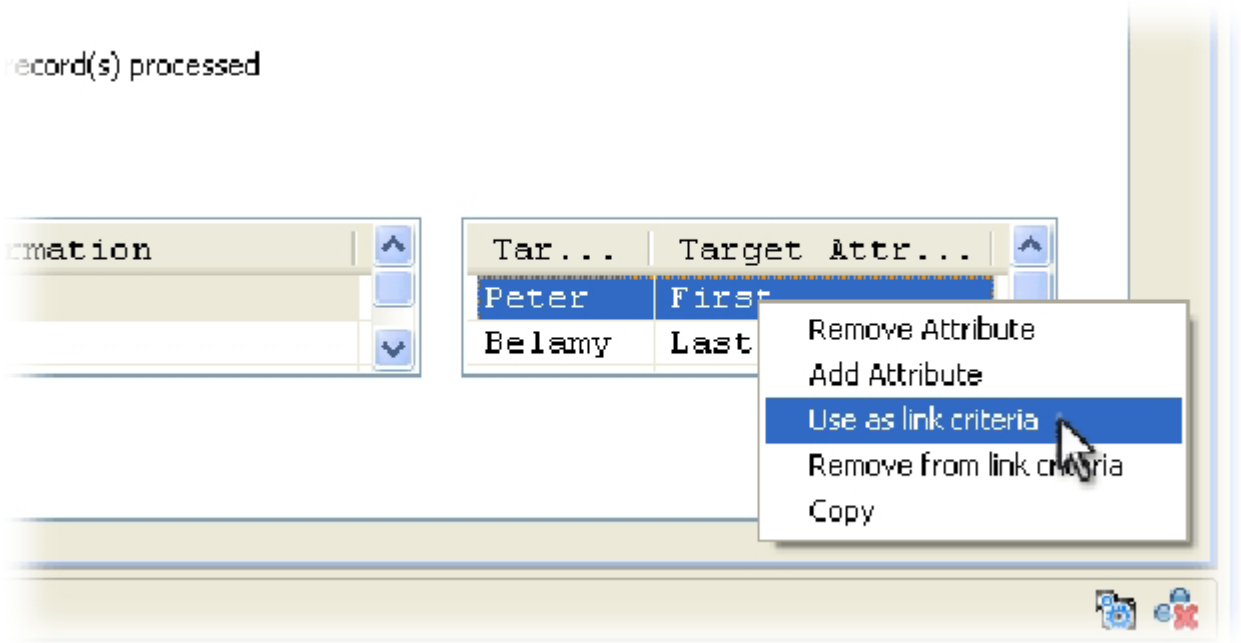

図 105. リンク基準の選択

出力コネクターがターゲットに書き込むときは、まず、指定されたリンク基準属性を使用してレコードを検索します。一致するものがない場合、新規の項目が追加されます。一致するものがあった場合は、そのレコードが更新されます。

これは、つまり、入力ソースと出力ターゲットの間で継続的に同期を行うように ETL ジョブが構成されたということです。

ETL ジョブの実行とスケジューリングのためのコマンド行資産

ETL AssemblyLine のデプロイメントの準備ができたら、ナビゲーターでプロジェクトを右クリックし、「**このプロジェクト/AssemblyLine をコマンド行から実行する**

47. これは、スナップショットを識別するために使用する属性なので、ある項目のこの値を変更すると、それはデルタ・エンジンにとっては*新規*のレコードのように見えます。

ために必要なファイルを作成します」オプションを選択します。

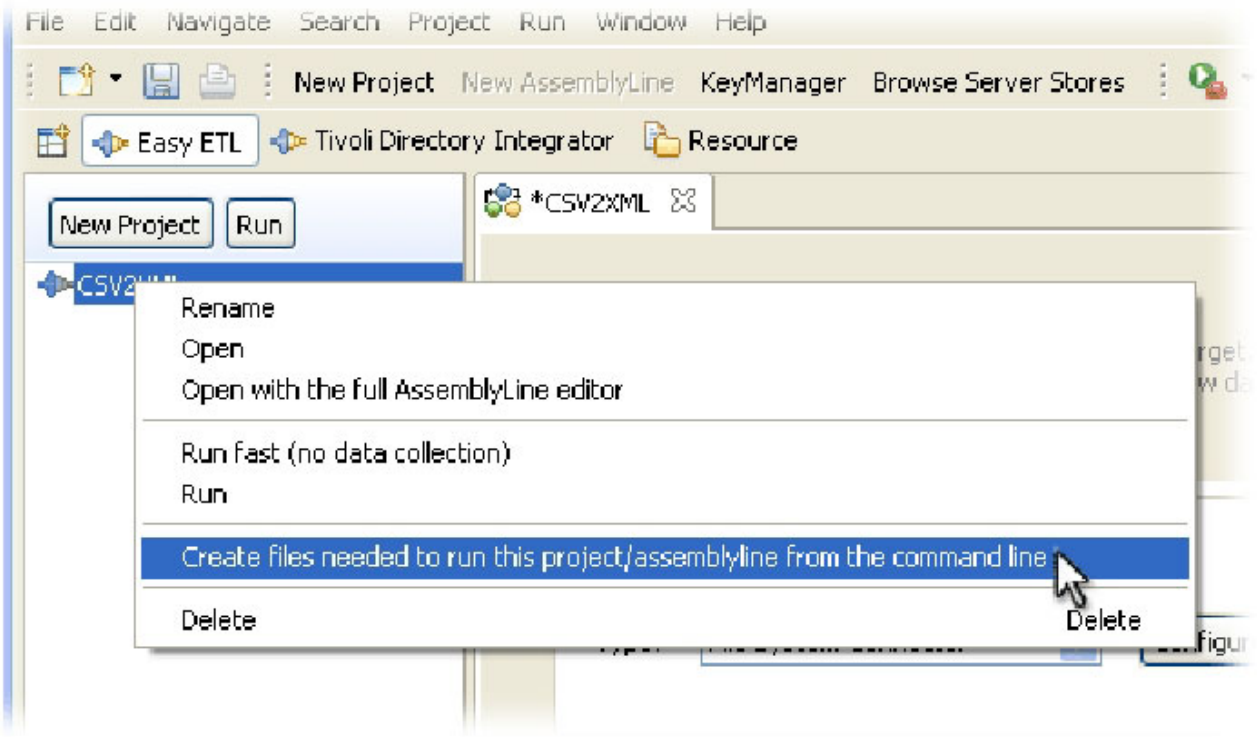

図 106. ETL ジョブを実行するためのコマンド行資産の作成

これにより、「エクスポート・ファイル」ダイアログが表示されます。ここでこのスクリプト/バッチ・ファイルを書き込みます。ファイル名はプロジェクトと同じ名前になるのでご注意ください。つまり、Windows 上で実行中のこのチュートリアルの演習では「`CSV2XML.bat`」になります。EasyETL プロジェクトをコマンド行から実行すると、ソリューションの最大のパフォーマンスを引き出せます。

同じ場所に XML ファイルも作成されます。これは、IBM Security Directory Integrator *構成* ファイルと呼ばれ、EasyETL AssemblyLine の詳細が含まれています。これは、IBM Security Directory Integrator サーバーが EasyETL AssemblyLine を実行するのに必要です。生成されたスクリプトをテキスト・エディターで開くと、1 行のコマンドが表示されます。これは、IBM Security Directory Integrator サーバーを開始し、そのサーバーに構成を参照させ、実行する AssemblyLine を指定するために必要です。この後は、このスクリプトを定期的に呼び出すスケジュールされたタスクまたは cronjob をセットアップすればよく、これによって、同期/マイグレーションのサービスが機能するようになります。

追加オプション

高速 ETL

データ・コレクターは強力なツールですが、ETL AssemblyLine はデータ収集および画面上の表示のために実行が遅くなります。EasyETL AL の処理をできるだけ速くしたい場合は、プロジェクトを選択して、ナビゲーターの上部の「実行します」ボタンを押すか、プロジェクトを右クリックして「**高速**

実行 (データ収集なし)」オプションを選択します。

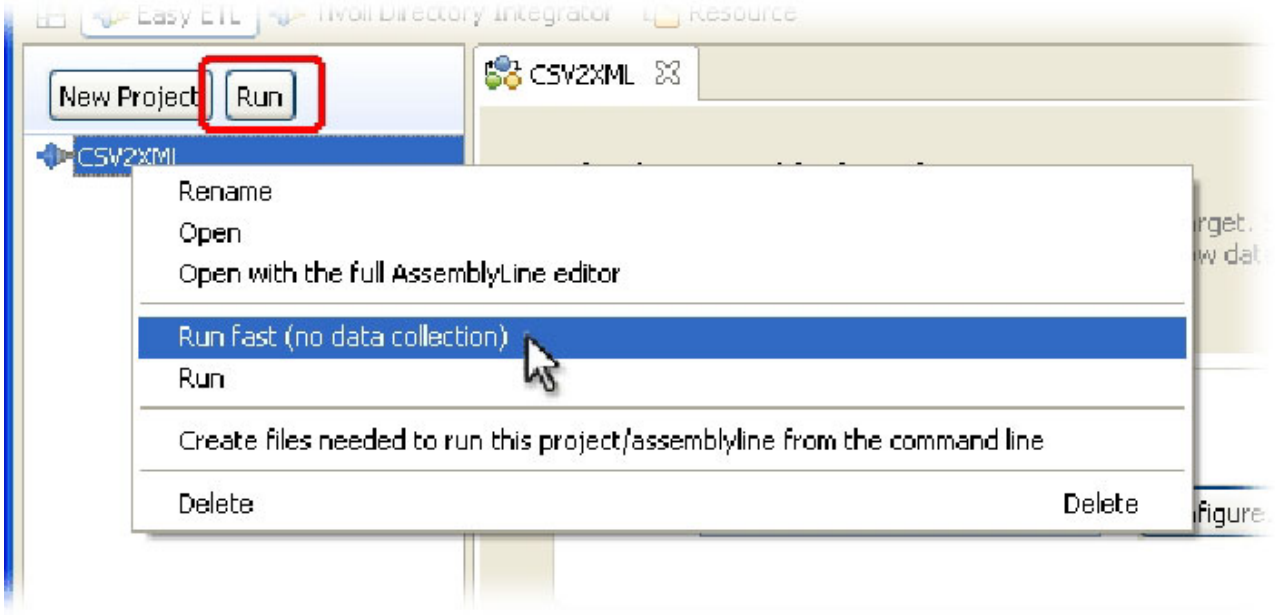

図 107. ETL ジョブのフルスピード実行

どちらのオプションでも、AL が高速で実行されると、コンソール・ディスプレイが開き、AssemblyLine からのログ・メッセージがそこに表示されます。

プロジェクトのコンテキスト・メニューの「**実行します**」オプションは、データ収集ありで ETL ジョブを実行するのでご注意ください。

入力データ・セットのフィルター処理

もう 1 つの強力な機能は、入力データ・セットの内容を制御する能力です。これは、入力ソースがデータベースまたはディレクトリーの場合に使用可能です。

例えば、入力に「LDAP コネクター」を選択し、このコンポーネントの構成ダイアログを確認します。「**検索フィルター**」パラメーターの隣に 3 つのドット (...) のラベルが付いたボタンがあります。これを押すとリンク基準エディターがオープンされるので、ここで、このコネクターが読み取る結果セットを構築する際に適用される検索ルールを定義することができます。

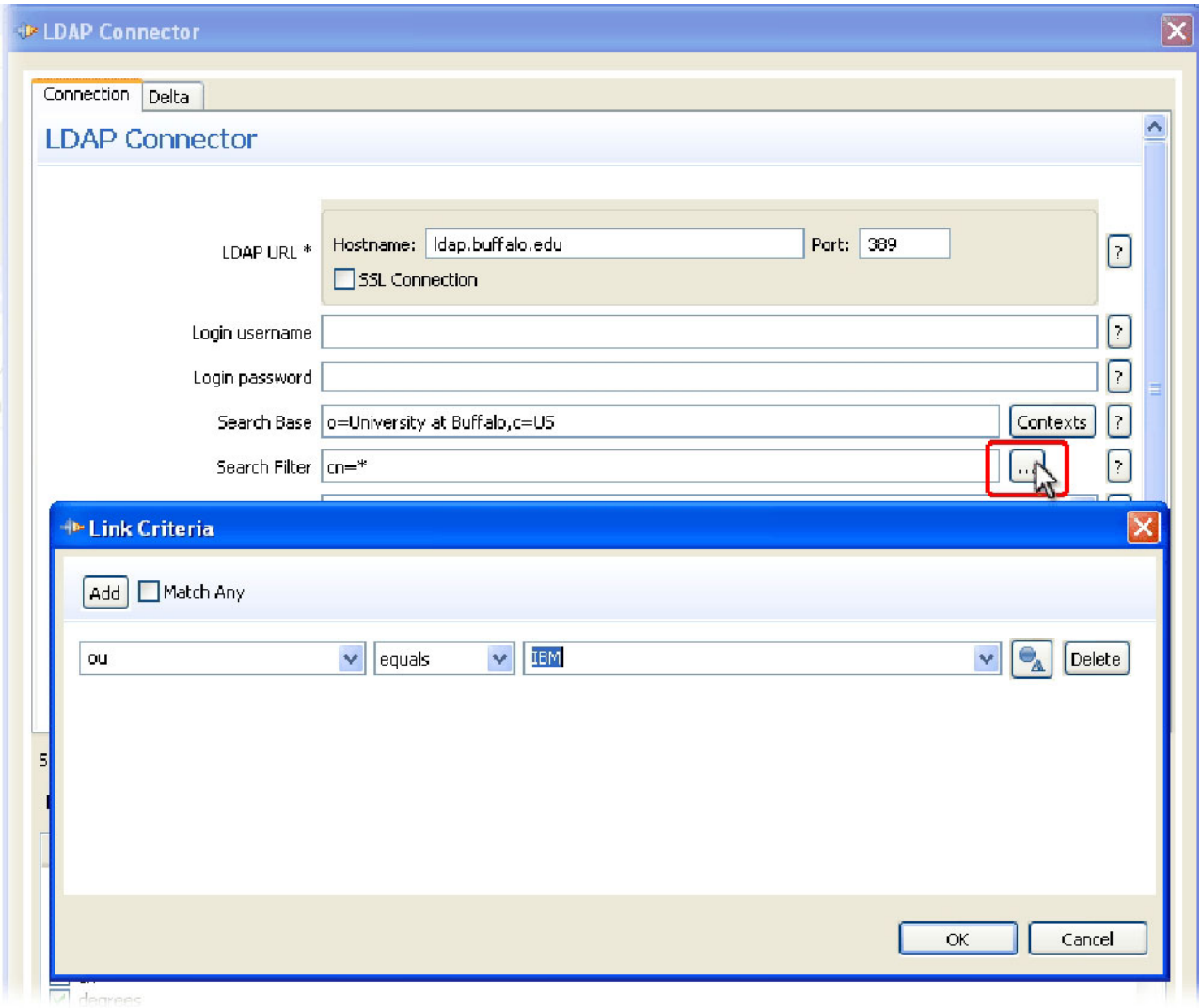

図 108. 入力コネクターのリンク基準の定義

これと同じ機能がデータベースおよび JDBC コネクターで使用可能です。このとき、「**選択**」パラメーターの隣に (...) ボタンが表示されます。

LDAP 検索構文を直接自分で検索パラメーターに入力することができますが、その場合、LDAP 検索フィルターまたは JDBC 検索文の構文を知らなければなりません。しばしば、リンク基準を使用して、基礎となる構文をコネクターに処理させることで、必要な表現を選択する方がより簡単です。

EasyETL AssemblyLine のレベルアップ

ETL プロジェクトをフル機能の IBM Security Directory Integrator AssemblyLine エディターで開くと、カスタム・ロギングおよび監査、エラー処理、フェイルオーバー・ロジック、自動再接続、データ拡大 (結合) などを、マイグレーションまたは同期ソリューションに追加することができます。これには、プロジェクトを右クリックし、「**フル機能の AssemblyLine エディターで開く**」オプションを選択します。まだ EasyETL ワークベンチで作業していますが、AssemblyLine で使用可能な追加機能を使用できます。

これがお好みに合い、使用してみようと思う場合は、Security Directory Integrator のパースペクティブに切り替えて (「**Windows**」 > 「**Open Perspective**」 > 「**Security Directory Integrator**」)、フル IBM Security Directory Integrator ワークベンチで作業を開始してください。これで EasyETL はマスターしました。今度はセクション 1 に戻って、 IBM Security Directory Integrator のすべての機能について掘り下げた学習を開始することをお勧めします。

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。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、 IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

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。

IBM の将来の方向性および指針に関するすべての記述は、予告なく変更または撤回される場合があります。これらは目標および目的を提示するものにすぎません。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

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。お客様は、IBM 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。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

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。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。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。

Adobe、PostScript は、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

IT Infrastructure Library は英国 Office of Government Commerce の一部である the Central Computer and Telecommunications Agency の登録商標です。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Centrino、Intel Centrino ロゴ、Celeron、Xeon、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子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。

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

Microsoft、Windows、Windows NT 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。

ITIL は英国 The Minister for the Cabinet Office の登録商標および共同体登録商標であって、米国特許商標庁にて登録されています。

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。

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。

Cell Broadband Engine は、Sony Computer Entertainment, Inc. の米国およびその他の国における商標であり、同社の許諾を受けて使用しています。

Linear Tape-Open、LTO、LTO ロゴ、Ultrium、および Ultrium ロゴは、HP、IBM Corp. および Quantum の米国およびその他の国における商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ラ行]

A

C

I

J

L

N

T

W

Printed in Japan

GI88-4235-03

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21