



DTL 変換の開発

Version 2023.1
2024-01-02

DTL 変換の開発

InterSystems IRIS Data Platform Version 2023.1 2024-01-02

Copyright © 2024 InterSystems Corporation

All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, および TrakCare は、InterSystems Corporation の登録商標です。HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, および InterSystems TotalView™ For Asset Management は、InterSystems Corporation の商標です。TrakCare は、オーストラリアおよび EU における登録商標です。

ここで使われている他の全てのブランドまたは製品名は、各社および各組織の商標または登録商標です。

このドキュメントは、インターシステムズ社(住所: One Memorial Drive, Cambridge, MA 02142)あるいはその子会社が所有する企業秘密および秘密情報を含んでおり、インターシステムズ社の製品を稼働および維持するためにのみ提供される。この発行物のいかなる部分も他の目的のために使用してはならない。また、インターシステムズ社の書面による事前の同意がない限り、本発行物を、いかなる形式、いかなる手段で、その全てまたは一部を、再発行、複製、開示、送付、検索可能なシステムへの保存、あるいは人またはコンピュータ言語への翻訳はしてはならない。

かかるプログラムと関連ドキュメントについて書かれているインターシステムズ社の標準ライセンス契約に記載されている範囲を除き、ここに記載された本ドキュメントとソフトウェアプログラムの複製、使用、廃棄は禁じられている。インターシステムズ社は、ソフトウェアライセンス契約に記載されている事項以外にかかるソフトウェアプログラムに関する説明と保証をするものではない。さらに、かかるソフトウェアに関する、あるいはかかるソフトウェアの使用から起こるいかなる損失、損害に対するインターシステムズ社の責任は、ソフトウェアライセンス契約にある事項に制限される。

前述は、そのコンピュータソフトウェアの使用およびそれによって起こるインターシステムズ社の責任の範囲、制限に関する一般的な概略である。完全な参照情報は、インターシステムズ社の標準ライセンス契約に記載され、そのコピーは要望によって入手することができる。

インターシステムズ社は、本ドキュメントにある誤りに対する責任を放棄する。また、インターシステムズ社は、独自の裁量にて事前通知なしに、本ドキュメントに記載された製品および実行に対する代替と変更を行う権利を有する。

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

目次

1 DTL ツールの概要	1
1.1 背景	1
1.2 データ変換ビルダ・ページの概要	1
1.3 DTL ダイアグラムの概要	2
1.4 表示の制御	3
1.5 データ変換リスト・ページの概要	4
1.6 その他のツール	4
1.7 データ変換の使用法	4
2 データ変換の作成	7
2.1 変換の作成	7
2.2 既存の変換のオープン	8
2.3 変換詳細の指定	8
2.3.1 [作成] の [既存] オプションの使用法	9
2.4 変換アクションの編集	10
2.4.1 アクションの追加	10
2.4.2 アクションの編集	10
2.5 アクションの再配置	11
2.6 アクションのグループの操作	11
2.7 変更の取り消し	12
2.8 変換の保存	12
2.9 変換のコンパイル	13
2.10 変換の削除	13
3 構文ルール	15
3.1 メッセージ・プロパティへの参照	15
3.2 リテラル値	15
3.2.1 XML 予約文字	16
3.2.2 仮想ドキュメント内のセパレータ文字	16
3.2.3 XML 予約文字がセパレータでもある場合	17
3.2.4 数字のコード	17
3.3 有効な式	17
4 Assign アクションの追加	19
4.1 概要	19
4.1.1 オブジェクトとオブジェクト参照	19
4.2 ソース・メッセージのコピー	19
4.3 ソース・プロパティからターゲット・プロパティへの値のコピー	20
4.4 すべてのサブプロパティの値のコピー	20
4.5 ターゲット・プロパティへのリテラル値の割り当て	21
4.6 ターゲット・プロパティの値に対する式の使用	22
4.6.1 データ変換関数ウィザードの使用法	22
4.7 コレクション項目への値の割り当て	23
4.8 リスト項目の挿入	24
4.9 リスト項目の追加	25
4.10 コレクション項目の削除	26
4.11 コレクション・プロパティの消去	27
5 その他のアクションの追加	29

5.1 If アクションの追加	29
5.2 For Each アクションの追加	30
5.2.1 For Each アクションのショートカット	32
5.2.2 ターゲット・コレクションのアンロード	32
5.2.3 ラージ・メッセージでの〈STORE〉エラーの回避	32
5.3 Subtransform アクションの追加	33
5.4 Trace アクションの追加	34
5.5 Code アクションの追加	34
5.5.1 DTL 内でカスタム・コードを使用する場合のガイドライン	35
5.6 SQL アクションの追加	35
5.6.1 DTL 内で SQL を使用する場合のガイドライン	35
5.7 Switch アクションの追加	36
5.8 Case アクションの追加	36
5.9 Default アクションの追加	36
5.10 Break アクションの追加	36
5.11 Comment アクションの追加	37
6 データ変換のテスト	39
6.1 変換テスト・ページの使用法	39
6.2 プログラムによる変換のテスト	40

1

DTL ツールの概要

この章では、DTL 変換を開発してテストするために InterSystems IRIS® から提供されているツールについて説明します。

1.1 背景

データ変換では、別のメッセージを変換して新しいメッセージが作成されます。プロダクションでデータ変換を使用する主な目的は、ターゲット・システムの要件に合わせて送信メッセージを調整することです。

DTL 変換は、管理ポータルかスタジオのどちらかで使用可能な DTL エディタで視覚的に作成して編集できます。DTL エディタは非技術系ユーザ向けのエディタです。DTL という用語は Data Transformation Language (データ変換言語) の頭字語であり、このエディタで作成された変換の定義を表現するために InterSystems IRIS 内部で使用される XML ベースの言語です。

データ変換は、ビジネス・プロセス、別のデータ変換、またはビジネス・ルールから呼び出すことができます。ビジネス・プロセス、データ変換、およびビジネス・ルールで使用可能なオプションが重複していることに注意してください。違いを確認する場合は、“プロダクションの開発”の“[ビジネス・ロジック・ツールの比較](#)”を参照してください。[データ変換を作成](#)して、自分自身でこれらのツールを使用してみることもできます。

1.2 データ変換ビルダ・ページの概要

[[データ変換ビルダ](#)] ページを使用すると、DTL 変換を作成、編集、およびコンパイルすることができます。

管理ポータルでこのページにアクセスするには、[[Interoperability](#)]→[[ビルド](#)]→[[データ変換](#)]を選択します。

ページが開いて、このネームスペースで最後に開かれた変換 (存在する場合) が表示されます。このページは以下の領域で構成されています。

- ・ 最上部にあるリボン・バーには、DTL 変換を作成して開いたり、表示されている変換をコンパイルしたり、ダイアグラムのズーム表示を変更したりするためのオプションが表示されます。
これらのオプションの詳細は、“[データ変換の作成](#)”の章を参照してください。
- ・ 左側の領域の上部には DTL ダイアグラムが表示されます。[次の節](#)でこの領域について詳しく説明します。
- ・ 左側の領域の下部には DTL 変換内で定義されたアクションの一覧表が表示されます。InterSystems IRIS がこの変換を使用するときは、これらのアクションをここに列挙された順に実行します。
- ・ 右側の領域には次の 3 つのタブが表示されます。

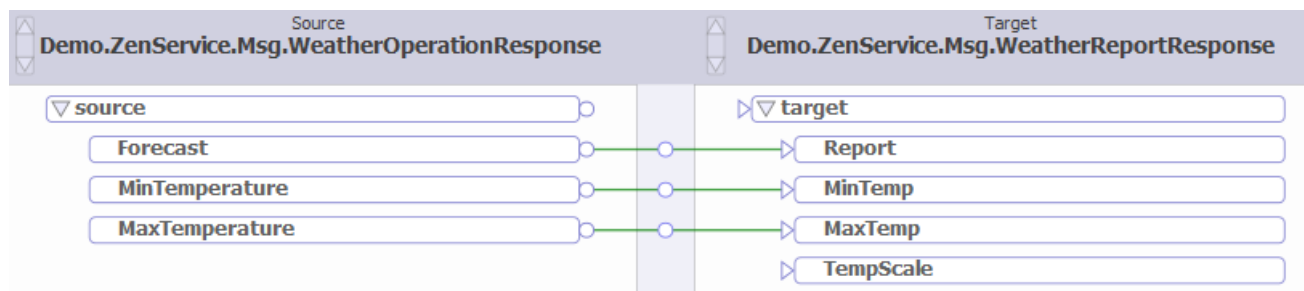
- [変換] - 変換に関する情報を編集できます。詳細は、後述する“[変換詳細の指定](#)”を参照してください。
- [アクション] - 選択したアクションの詳細を編集できます。後半の章で、[assign アクション](#)や[その他の種類のアクション](#)について詳しく説明します。
- [ツール] - 表示されている変換を[テスト](#)するためのウィザードを起動できます。詳細は、後述する“[データ変換のテスト](#)”を参照してください。

- ・ これらの 3 つの領域のサイズを変更できます。

Tip ヒン スタジオで DTL 変換クラスを開くと、このページと同様のページが表示されます。スタジオのデータ変換ビルダ・ト ページで [他のコードを表示] を選択することによって、データ変換の XML 定義を表示して編集できます。

1.3 DTL ダイアグラムの概要

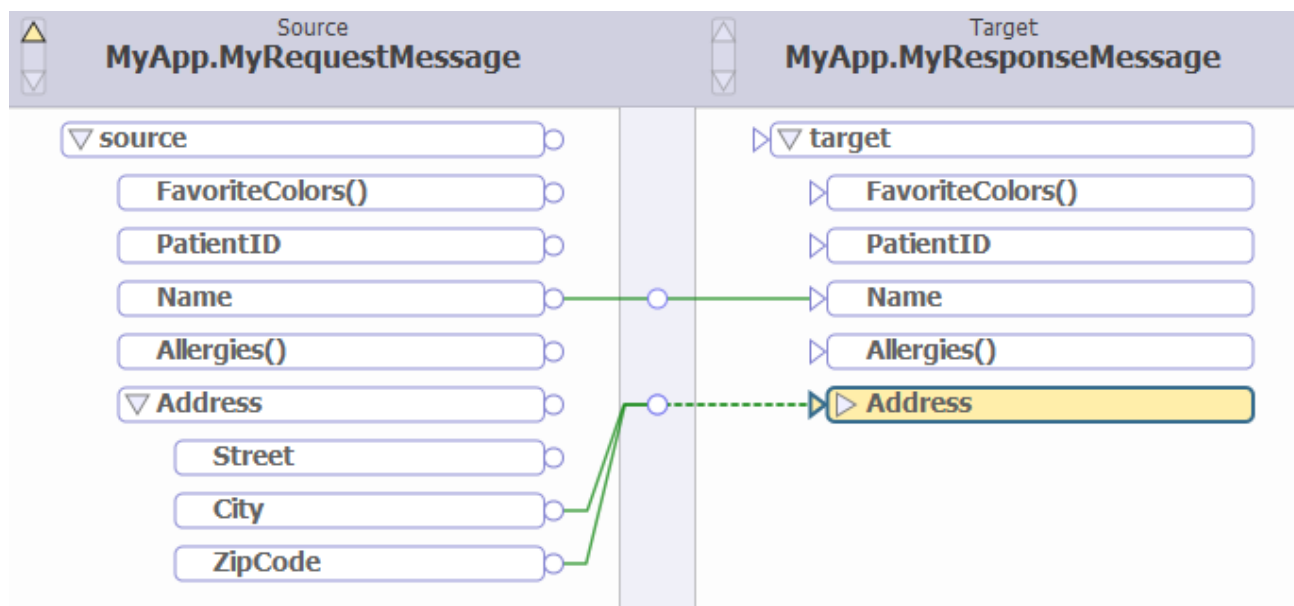
DTL クラスの DTL ダイアグラムを以下に示します。



以下の点に注意してください。

- ・ 左側の領域には、ソース・メッセージが表示されます。列の上のヘッダにはソース・メッセージ・クラスの名前が、列内のボックスにはソース・メッセージのプロパティが表示されます。
- ・ 右側の領域には、同様にターゲット・メッセージが表示されます。
- ・ 上側の領域には、上記領域ごとにスクロール・ボタンが表示されます。
- ・ ダイアグラムには、変換内部のアクションを表すコネクタが表示されます。ここに表示されたアクションは、ソース・プロパティからターゲット・プロパティに値をコピーします。
- ・ 中央の仕切り (青色の列) には、各コネクタ線上のアイコンが表示されます。これらのアイコンの目的は、コネクタの選択を容易にすることです (コネクタ線はそのどこをクリックしても選択できますが、この中央の仕切りに表示されたアイコンをクリックの方が簡単です)。

以下にもう 1 つの例を示します。



この例では、ソース・クラスとターゲット・クラスがもう少し複雑になります。以下の点に注意してください。

- ・ **FavoriteColors** プロパティは、文字列のリストとして定義されています。ここでは、このプロパティの名前の最後に () が付けられています。

この例の **Allergies** はもう 1 つのコレクション・プロパティです。

- ・ **Address** プロパティは、**Street**、**City**、および **ZipCode** の各プロパティを含むオブジェクトとして定義されています。このプロパティのボックス内部に三角形が表示されていることに注目してください。

左側の列では、このプロパティが展開モードで表示されているため、プロパティを確認できます。ボックス内の三角形は塗りつぶされておらず、下を向いています。

右側の列では、このプロパティが折りたたみモードで表示されています。ボックス内の三角形は塗りつぶされており、右を向いています。

- ・ **Address** プロパティに関して、**Address** が折りたたまれている側でコネクタが点線で表示されています。これは、この側で assign アクションのサブプロパティが非表示になっていることを示しています。

1.4 表示の制御

複数の方法でデータ変換ビルダ・ページの表示を制御できます。

- ・ リボン・バーで、**[表示]** オプションをクリックできます。



このボタンを使用すれば、変換ダイアグラムとアクション・リストの両方をページの左ペインに表示することも、表示したくないセクションを折りたたむこともできます。

- ・ リボン・バーのドロップダウン・リストからズーム・オプションを選択できます。デフォルトで、このリストには **100%** と表示されます。リスト内の値をクリックすることで、DTL ダイアグラムのサイズを縮小または拡大できます。
- ・ [前の節](#) で説明したように、DTL ダイアグラムのヘッダ領域にあるスクロール・バーを使用します。
- ・ [前の節](#) で説明したように、プロパティの表示を折りたたんだり、展開したりします。

1.5 データ変換リスト・ページの概要

[Interoperability]→[リスト]→[データ変換] ページには、現在のネームスペースで定義されているデータ変換クラスのリストが表示されます。

このページには、次の 2 種類の変換が表示されます。

- ・ DTL 変換は青色で表示されます。いずれかの変換をダブルクリックすることで、[データ変換ビルダ](#)で開くことができます。
- ・ カスタム変換は黒色で表示されます。これらのクラスは **Ens.DataTransform** に基づいており、DTL を使用しません。これらの編集は、スタジオなどのサポートされている IDE で行う必要があります。

このページを使用するには、変換クラスを選択してから、リボン・バーで以下のコマンドのいずれかをクリックします。

- ・ **[編集]** – (DTL 変換のみ) [データ変換ビルダ](#)を使用してデータ変換を変更または表示できます。
- ・ **[テスト]** – 変換テスト・ウィザードを使用して選択した変換クラスをテストできます。
詳細は、後述する“[データ変換のテスト](#)”の章を参照してください。
- ・ **[削除]** – 選択した変換クラスを削除できます。
- ・ **[エクスポート]** – 選択した変換クラスを XML ファイルにエクスポートできます。
- ・ **[インポート]** – XML ファイルにエクスポートされたデータ変換をインポートできます。

InterSystems IRIS の他のクラスと同様に、これらのクラスもエクスポートまたはインポートできます。管理ポータルの **[システムエクスプローラ]**→**[グローバル]** ページを使用することも、スタジオの **[ツール]** メニューにある **[エクスポート]** コマンドと **[インポート]** コマンドを使用することもできます。

1.6 その他のツール

データ変換はプログラムから呼び出すこともできるため、テストする場合に便利です。詳細は、“[データ変換のテスト](#)”の章を参照してください。

また、データ変換はクラスであるため、他のクラスと同様に、編集したり、操作したりできます。

1.7 データ変換の使用法

データ変換は以下のプロダクションの部分から呼び出すことができます。

- ・ 別の DTL データ変換から。“[その他のアクションの追加](#)”の章の“[Subtransform アクションの追加](#)”を参照してください。
- ・ BPL ビジネス・プロセスから。“ビジネス・プロセス言語およびデータ変換言語リファレンス”の“[<transform>](#)”を参照してください。
- ・ ビジネス・ルールから。“ビジネス・ルールの開発”の“[send アクションの変換とターゲットの選択](#)”を参照してください。
- ・ カスタム・ビジネス・プロセスまたはカスタム DTL 変換から。これを実現するには、“[データ変換のテスト](#)”の章で説明されているように、プログラムから呼び出します。

注釈 この節は、DTL 変換とカスタム変換の両方に適用されます。

2

データ変換の作成

この章では、データ変換の作成方法と編集方法について簡単に説明します。

後半の章では、データ変換で使用する[構文](#)、[assign アクション](#)の詳細、および[その他の種類のアクション](#)の詳細について説明します。

オンライン学習に移動して、ご自身で[データ変換を作成](#)してみることもできます。

重要 非アクティブ状態が続くと、インターシステムズの管理ポータルによってログアウトされ、保存されていない変更が破棄される可能性があります。非アクティブ状態の時間は、InterSystems IRIS サーバの呼び出しから次の呼び出しまでです。すべてのアクションがサーバの呼び出しを引き起こすわけではありません。例えば、**[保存]** をクリックした場合はサーバが呼び出されますが、テキスト・フィールドに入力した場合はサーバの呼び出しは生じません。このため、データ変換を編集していて、**[セッションタイムアウト]** のしきい値より長い間 **[保存]** をクリックしないと、セッションは期限切れとなり、保存されていない変更は破棄されます。ログアウト後、ログイン・ページが表示されるか、現在のページが更新されます。詳細は、“[管理ポータルの自動ログアウト動作](#)” を参照してください。

2.1 変換の作成

変換を作成するには：

1. **[次]** をクリックします。

変換を表示して変更を加えたが、まだ保存していない場合は、InterSystems IRIS® から先に進む (変更を破棄する) かどうかの確認が求められます。

その後で、変換に関する基本情報を指定可能なダイアログ・ボックスが表示されます。

2. 以下の情報の一部または全部を指定します。

- ・ **[パッケージ]** (必須) – パッケージ名を入力するか、矢印をクリックして現在のネームスペース内のパッケージを選択します。
予約パッケージ名は使用しないでください。“プロダクションの開発” の “[予約パッケージ名](#)” を参照してください。
- ・ **[名前]** (必須) – データ変換クラスの名前を入力します。
- ・ **[説明]** – データ変換の説明を入力します。これがクラスの説明になります。
- ・ **[ソースタイプ]** と **[ソースクラス]** – この変換が入力として受け取るメッセージのタイプを指定します。
以下のいずれかを選択します。

- **[全メッセージ]** – この変換を任意の入力メッセージ・タイプで使用できます。
- **[X12]** – 入力メッセージは **EnsLib.EDI.X12.Document** のインスタンスです。
- **[EDIFACT]** – 入力メッセージは **EnsLib.EDI.EDIFACT.Document** のインスタンスです。
- **[XML]** – 入力メッセージは **EnsLib.EDI.XML.Document** のインスタンスです。

または、**[ソースクラス]** の検索アイコンをクリックしてから、クラスを選択します。

- ・ **[ソースドキュメントタイプ]** (メッセージが仮想ドキュメントの場合にのみ適用可能) – ソース・メッセージのドキュメント・タイプを入力または選択します。このネームスペースにロードされた該当するスキーマで定義された任意のタイプを選択できます。
- ・ **[ターゲットタイプ]** と **[ターゲットクラス]** – この変換が出力として生成するメッセージのタイプを指定します。**[ソースタイプ]** と **[ソースクラス]** の選択肢を参照してください。
- ・ **[ターゲットドキュメントタイプ]** (メッセージが仮想ドキュメントの場合にのみ適用可能) – ターゲット・メッセージのドキュメント・タイプを入力または選択します。このネームスペースにロードされた該当するスキーマで定義された任意のタイプを選択できます。

[パッケージ] と **[名前]** を除いて、後からこれらの詳細を編集できます。

3. **[変換]** タブで詳細を指定します。“[変換詳細の指定](#)”を参照してください。

2.2 既存の変換のオープン

変換を開くには:

1. **[開く]** をクリックします。
変換を表示して変更を加えたが、まだ保存していない場合は、InterSystems IRIS から先に進む (変更を破棄するか) かどうかの確認が求められます。
2. 変換を含むパッケージをクリックします。
次に、必要に応じて、サブパッケージをクリックします。
3. 変換クラスをクリックします。
4. **[OK]** をクリックします。

2.3 変換詳細の指定

変換の場合は、**[変換]** タブに変換全体に適用される詳細が表示されます。詳細の一部が事前に指定されている場合とされていない場合があります。その他の詳細はここでしか編集できません。このような詳細を以下に示します。

- ・ **[名前]** (読み取り専用) – データ変換クラスの完全なパッケージ名とクラス名。
- ・ **[作成]** – 変換でターゲット・メッセージを作成する方法を指定します。以下のいずれかを選択します。
 - **[新規]** – データ変換内の要素を実行する前に、ターゲット・クラス (および該当する場合のタイプ) の新しいオブジェクトを作成します。これがデフォルトです。
 - **[コピー]** – 変換内の要素を実行する前に、ターゲット・オブジェクトとして使用するソース・オブジェクトのコピーを作成します。

- **【既存】** – データ変換の呼び出し側から指定された既存のオブジェクトをターゲット・オブジェクトとして使用します。[次の項](#)を参照してください。
- **【ソースクラス】** – この変換が入力として受け取るメッセージのタイプを指定します。詳細は、前述した“[変換の作成](#)”を参照してください。
- **【ソースドキュメントタイプ】** (メッセージが仮想ドキュメントの場合にのみ適用可能) – ソース・メッセージのドキュメント・タイプを指定します。
- **【ターゲットクラス】** – この変換が出力として生成するメッセージのタイプを指定します。詳細は、前述した“[変換の作成](#)”を参照してください。
- **【ターゲットドキュメントタイプ】** (メッセージが仮想ドキュメントの場合にのみ適用可能) – ターゲット・メッセージのドキュメント・タイプを指定します。
- **【言語】** – この DTL 内のすべての式で使用する言語を指定します。これは **objectscript** である必要があります。
- **【レポートエラー】** – この変換の実行中に発生したエラーをログに記録するかどうかを指定します。このオプションを選択した場合は、エラーが警告としてイベント・ログに書き込まれます。また、すべてのエラーを戻り値として含めている複合ステータス・コードが返されます。このオプションはデフォルトで選択されています。
- **【存在しないソースセグメントとプロパティを無視する】** – 仮想ドキュメントまたはオブジェクトのプロパティの存在しないソース・セグメントからフィールド値を取得しようとして発生したエラーを無視するかどうかを指定します。このオプションを選択した場合は、これらのエラーが抑制され、指定されたソースが存在しないサブ変換は呼び出されません。このオプションはデフォルトで選択されています。

テストと条件付きロジック分岐を含めて、必要な要素が存在することを確認することにより、この動作を正確に制御できます。

- **【空の繰り返しフィールドをヌルとして扱う】** – InterSystems IRIS が空の繰り返しフィールドに対する以下のアクションをスキップするかどうかを指定します。
 - **【foreach アクション】** – このオプションを選択すると、InterSystems IRIS は、空の繰り返しフィールドに対して foreach アクションを実行しません。
 - **【assign アクション】** – このオプションを選択すると、source と target の両方のフィールドが繰り返しフィールドであることを示すショートカット表記を使用していて、source フィールドが空の場合、InterSystems IRIS はこれらの繰り返しフィールドに対して assign アクションを実行しません。例えば、`source.{PV1:AdmittingDoctor()}` フィールドが空の場合にこのオプションを選択すると、InterSystems IRIS は以下のアクションを実行しません。

```
<assign value='source.{PV1:AdmittingDoctor()}'
property='target.{PV1:AdmittingDoctor()}' action='set'
```

ただし、target フィールドは繰り返しフィールドではないため、InterSystems IRIS は、以下のよく似たアクションは実行します。

```
<assign value='source.{PV1:AdmittingDoctor()}'
property='target.{PV1:AdmittingDoctor(1)}' action='set' />
```

このオプションは既定では無効です。

- **【説明】** – データ変換の説明。

2.3.1 【作成】の【既存】オプションの使用方法

【作成】の【既存】オプションを使用すると、ターゲットを既存のオブジェクトとして指定できるため、パフォーマンスが向上します。このオプションは、一連の変換をプログラムから呼び出す（またはその他のシーケンシャル処理を実行する）場合に適用されます。次のようなシナリオで使用されます。

- ・ 順番に実行する 3 つの変換があります。
 1. MyApp.ADTTransform – [作成] の [新規] オプションを使用します。
 2. MyApp.MRNTTransform – [作成] の [既存] オプションを使用します。
 3. MyApp.LabXTransform – [作成] の [既存] オプションを使用します。
- ・ 次のように変換を呼び出します。

```
do MyApp.ADTTransform.Transform(message, .target)
do MyApp.MRNTTransform(target, .newtarget)
do MyApp.LabXTransform(newtarget, .outmessage)
```

2.4 変換アクションの編集

この節では、変換内のアクションの追加方法と編集方法について簡単に説明します。この節は以下の項で構成されています。

- ・ [アクションの追加](#)
- ・ [アクションの編集](#)

後半の章では、データ変換で使用する[構文](#)、[assign アクション](#)の詳細、および[その他の種類のアクション](#)の詳細について説明します。

2.4.1 アクションの追加

アクションを追加するには、次の手順を実行します。

1. オプションで、追加するアクションの種類に応じて、ソースまたはターゲット・プロパティをクリックします。
2. リボン・バーの **[アクション追加]** ドロップダウン・リストからアクションを選択します。
3. **[アクション]** タブでこのアクションの詳細を編集します。

該当する場合は、選択したプロパティが **[プロパティ]** フィールドに表示され、基準として使用されます。必要に応じて、**[無効]** チェック・ボックスを使用してこのアクションを無効にできます。foreach アクションまたは if アクションを無効にした場合は、そのブロック内のすべてのアクションも無効になります。

[後述](#)するように、assign アクションに関するその他のテクニックを使用できます。

2.4.2 アクションの編集

アクションを編集するには、まず、それを選択します。そのためには、以下のように操作します。

- ・ ダイアグラムの下でテーブルで対応する行をクリックします。
- ・ ダイアグラムでアクションを表示する場合は、中央の仕切りにある該当するコネクタ線上のアイコンをクリックします。

この項目をクリックすると、その色が変わり、コネクタ線が太くなり、ソース・プロパティとターゲット・プロパティは色付きでハイライト表示されます。これは、下の図のように、コネクタが選択されていることを意味します。



ここで、[アクション] タブ上の値を編集します。必要に応じて、[無効] チェック・ボックスを使用してこのアクションを無効にできます。foreach アクションまたは if アクションを無効にした場合は、そのブロック内のすべてのアクションも無効になります。







Tip ヒン ダイアグラム内のプロパティをダブルクリックすると、該当する場合に、現在選択されているアクションが更新されます。ソース内のフィールドをダブルクリックすると、エディタは選択されたアクションの値の設定が必要であると解釈します。同様に、ターゲット・フィールドをダブルクリックすると、エディタは選択したアクションのターゲットの設定が必要であると解釈します。

2.5 アクションの再配置

アクションは、ダイアグラムの下テーブルに表示された順に実行されます。

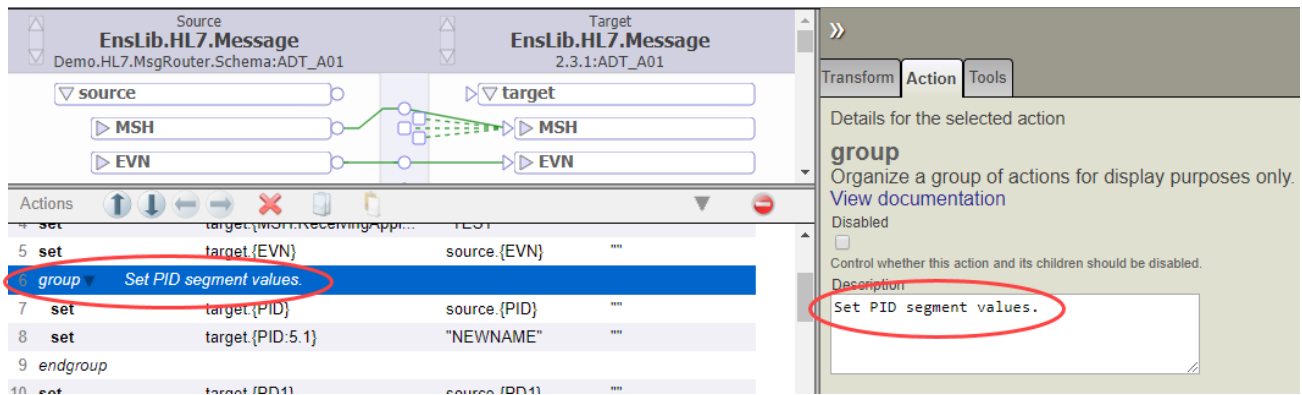
アクションを並べ替えるには、次のように、DTL ダイアグラムの下テーブルを使用する必要があります。

1. そのアクションに対応する行をクリックします。
2. 必要に応じて、その行内の以下のアイコンのいずれかをクリックします。

ツール	説明
	選択したアクションを 1 つ上の位置に移動します。そのアクションがグループ (for each ブロックや if ブロックなど) の最初のアクションである場合は、上に移動してグループの外に配置されます。
	選択したアクションを 1 つ下の位置に移動します。そのアクションがグループの最後のアクションである場合は、グループの直後に移動します。例えば、そのアクションが if ブロックの最後のアクションである場合、ブロックの直後に移動します。そのアクションが else の直前の if ブロック内の最後のアクションの場合は、else ブロック内の最初の位置に移動します。
	選択されたアクションを現在のグループ (for each ブロックや if ブロックなど) の外に移動します。これにより、アクションは現在のグループからそのグループの直前の位置に移動します。
	選択されたアクションを次のアクションのグループ (for each ブロックや if ブロックなど) に移動します。
	データ変換のすべてのアクションを削除します。
	この行のアクションを削除します。

2.6 アクションのグループの操作

group アクションを使用して、複数のアクションを 1 つの表示グループにまとめることができます。アクションをグループ化すると、ダイアグラムの下テーブルで簡単に編成できます。グループを識別できるように、[アクション] タブで定義した説明がリストに表示されます。



アクションをグループの中または外に移動するには、リスト内でアクションを選択し、 (グループに移動) または (グループから移動) をクリックします。リストをよりわかりやすくするために、リストを確認するときにグループを折りたたんだり、展開したりすることができます。グループを折りたたんで、含まれているアクションを非表示にするには、アクション名の横にある ▼ をクリックします。グループを展開するには、▶ をクリックします。テーブルの【アクション】バーにある ▼ ボタンと ▶ ボタンを使用して、すべてのグループを一度に折りたたんだり、展開したりすることもできます。

Actions				
#	Action	Condition	Property	Value
1	group ▼			
2	set		target.SourceConfigNam...	source.SourceConfigNam...
3	set		target.AlertText	source.AlertText
4	endgroup			
5	set		target.AlertDestinatio...	..Lookup("AlertTable",...

注釈 if、for each、switch、および case のアクションによって作成されたアクションのブロックを展開したり、折りたたんだりすることもできます。

2.7 変更の取り消し

過去の変更を取り消すには、[元に戻す] ボタン をクリックします。

2.8 変換の保存

変換を保存するには、以下のいずれかを実行します。

- ・ [保存] をクリックします。
- ・ [名前を付けて保存] をクリックします。次に、新しいパッケージ、クラス名、および説明を指定して、[OK] をクリックします。
- ・ [コンパイル] をクリックします。このオプションは、変換を保存してからコンパイルします。

2.9 変換のコンパイル

変換をコンパイルするには、[コンパイル] をクリックします。このオプションは、変換を保存してからコンパイルします。

2.10 変換の削除

変換を削除するには、[Interoperability]→[リスト]→[データ変換] ページという別のページを使用する必要があります。

変換を削除するには：

1. その名前が表示された行をクリックします。
2. [削除] ボタンをクリックします。
3. [OK] をクリックしてこのアクションを確定します。

3

構文ルール

この章では、さまざまな DTL アクション内でプロパティを参照したり、式を作成したりするための構文ルールについて説明します。

3.1 メッセージ・プロパティへの参照

変換内のほとんどのアクションにおいて、ソースまたはターゲット・メッセージのプロパティを参照する必要があります。プロパティを参照するためのルールは、操作するメッセージの種類によって異なります。

- ・ 仮想ドキュメント以外のメッセージの場合は、次のような構文を使用します。

```
source.propertyname
```

または

```
source.propertyname.subpropertyname
```

ここで、propertyname はソース・メッセージ内のプロパティで、subpropertyname はそのプロパティのプロパティです。

メッセージにコレクション・プロパティが含まれている場合は、“[プロダクション内での仮想ドキュメントの使用法](#)”の“[繰り返しフィールドの特殊なバリエーション](#)”を参照してください。ここで紹介する情報の一部は、仮想ドキュメントと標準メッセージの両方に適用されます。

- ・ XML 仮想ドキュメント以外の仮想ドキュメントの場合は、“[プロダクション内での仮想ドキュメントの使用法](#)”の“[仮想プロパティ・パスに関する構文ガイド](#)”に記載された構文を使用します。次の項も参照してください。
- ・ XML 仮想ドキュメントについては、“[プロダクション内での XML 仮想ドキュメントのルーティング](#)”を参照してください。

3.2 リテラル値

値をターゲット・プロパティに代入するときに、リテラル値を指定することがよくあります。リテラル値は、trace アクション内の値のような他の場所にも適している場合があります。

リテラル値は次のどちらかです。

- ・ 数値リテラルはただの数字です。例えば、42.3 などです。

- 文字列リテラルは二重引用符で囲まれた文字列です。例えば、"ABD" などです。

注釈 この文字列には XML 予約文字を含めることができません。詳細は、“[XML 予約文字](#)”を参照してください。

仮想ドキュメントの場合は、この文字列に仮想ドキュメント形式で使用するセパレータ文字を含めることができません。“[仮想ドキュメント内のセパレータ文字](#)”と“[XML 予約文字がセパレータでもある場合](#)”を参照してください。

3.2.1 XML 予約文字

DTL 変換は XML ドキュメントとして保存されるため、XML 予約文字の代わりに XML エンティティを使用する必要があります。

この文字を含めるには...	次の XML エンティティを使用します...
>	>
<	<
&	&
'	'
"	"

例えば、値の Joe's "Good Time" Bar & Grill をターゲット・プロパティに代入するには、**[値]** を次のように設定します。

```
"Joe&apos;s &quot;Good Time&quot; Bar &amp; Grill"
```

InterSystems IRIS® では、エディタに入力されたテキストの周りに自動的に CData ブロックが配置されるため、この制限は code アクションと sql アクションの内部に適用されません(XML 標準では、CData ブロックの間に XML と解釈すべきではないテキストが入ります。そのため、そのブロックに予約文字を含めることができます)。

3.2.2 仮想ドキュメント内のセパレータ文字

ほとんどの仮想ドキュメント形式で、セグメント間、フィールド間、サブフィールド間などで特定の文字がセパレータとして使用されます。メッセージ内に値を設定するときにこれらの文字をリテラル・テキストとして含める必要がある場合は、代わりに、そのドキュメント形式に適切なエスケープ・シーケンス (存在する場合) を使用する必要があります。

これらの文字は該当するドキュメント内で文書化されます。詳細は、以下を参照してください。

- “[プロダクション内での EDIFACT ドキュメントのルーティング](#)” の参照節内の “[セパレータ](#)”
- “[プロダクション内での X12 ドキュメントのルーティング](#)” の参照節内の “[セパレータ](#)”

重要 データ変換では、セパレータ文字とエスケープ・シーケンスをソース・メッセージとターゲット・メッセージで分けることができます。InterSystems IRIS が、自動的に、変換の実行後に必要に応じて値を調整します。これは、ソース・メッセージに適用されるセパレータ文字とエスケープ・シーケンスだけを考慮すればいいことを意味します。

3.2.3 XML 予約文字がセパレータでもある場合

- ・ 文字(& など)がセパレータでそれをリテラル文字として含めたい場合は、仮想ドキュメント形式に適用されるエスケープ・シーケンスを使用します。
- ・ それ以外の場合は、“XML 予約文字”で示したような XML エンティティを使用します。

3.2.4 数字のコード

リテラル文字列には、10 進表現または 16 進表現を含めることができます。

文字列 `&#n;` は Unicode 文字を表します (n は Unicode 文字の 10 進数)。例えば、`é` は、揚音アクセント符号付きのラテン文字 e (é) を表します。

また、文字列 `&#xh;` も Unicode 文字を表します (h は 16 進の Unicode 文字番号)。例えば、`¿` は逆さの疑問符 (¿) を表します。

3.3 有効な式

値をターゲット・プロパティに代入するときに、データ変換用に選択した言語で式を指定できます。式は、if アクションの条件、trace アクション内の値、code アクション内の文などの他の場所でも使用します。

有効なすべての式を以下に示します。

- ・ [前の節](#)で説明したリテラル値。
- ・ 関数コール (InterSystems IRIS はビジネス・ルールやデータ変換で使用するための一連のユーティリティ関数を提供しています。詳細は、“ビジネス・ルールの開発”の“[プロダクションで使用するユーティリティ関数](#)”を参照してください)。InterSystems IRIS にはこれらのための[ウィザード](#)が用意されています。
- ・ “[プロパティへの参照](#)”で説明したプロパティへの参照。
- ・ ルールでパスされた `aux` 変数への参照。データ変換がルールから呼び出される場合、`aux` 変数で以下の情報を提供します。
 - `aux.BusinessRuleName`—ルールの名前。
 - `aux.RuleReason`—ルールが起動された理由。これは、ロギングで使用する名前と同じです。例えば、`'rule#1:when#1'` などの値です。RuleReason が 2,000 文字を超える場合は、2000 文字に省略されます。
 - `aux.RuleUserData`—ルールで 'RuleUserData' プロパティに割り当てられた値。'RuleUserData' の値は常に、最後に設定されていた値になります。
 - `aux.RuleActionUserData`—ルールの when 節または otherwise 節でプロパティ 'RuleActionUserData' に割り当てられた値。

データ変換がルールからではなくコードから直接呼び出される場合、コードは 3 番目のパラメータで予備のデータを渡すことができます。3 番目のパラメータが設定されていないコードからデータ変換が呼び出される可能性がある場合、DTL コードは \$ISOBJECT 関数を使用して `aux` 変数が if アクションのオブジェクトであることを確認する必要があります。

- ・ データ変換用に選択したスクリプティング言語の構文を使用してこれらを組み合わせた式。前述した“[変換詳細の指定](#)”を参照してください。以下に留意してください。
 - ObjectScript の場合、連結演算子は、以下のように `_` (アンダースコア) 文字です。

```
value=' "prefix"_source.{MSH:ReceivingApplication}_"suffix" '
```

- \$CHAR や \$PIECE などの便利な ObjectScript 文字列関数については、“ObjectScript リファレンス”を参照してください。
- 概要は、“ObjectScript の使用法”を参照してください。

4

Assign アクションの追加

この章では、DTL 変換にさまざまな assign アクションを追加する方法について詳しく説明します。

重要 仮想ドキュメントでは、データ変換内のエスケープ・シーケンスを手動で変更しないでください。これらは InterSystems IRIS® によって自動的に処理されます。

その他の種類のアクションの追加方法は、[次の章](#)を参照してください。

アクションのグループを操作する方法の詳細は、“[アクションのグループの操作](#)”を参照してください。

4.1 概要

assign アクションには、set、clear、remove、append、および insert の 5 種類があります。assign アクションを作成してから、そのタイプを変更できます。これを実現するには、[アクション] タブの [アクション] ドロップダウンで別の値を選択します。

InterSystems IRIS では、それぞれの assign アクションが [DTL ダイアグラム](#)内のコネクタ線で表現されます。

4.1.1 オブジェクトとオブジェクト参照

ソースとして別のオブジェクトの任意のオブジェクト・プロパティまたは最上位のソース・オブジェクトから assign する場合、ターゲットは、オブジェクト自体ではなくそのオブジェクトの複製されたコピーを受け取ります。これにより、オブジェクト参照の不注意な共有を避け、ユーザ自身が複製オブジェクトを生成する手間を省きます。

代わりに、ソースとターゲット間でオブジェクト参照を共有する場合は、ソースから中間一時変数に assign してから、その変数からターゲットに assign する必要があります。

4.2 ソース・メッセージのコピー

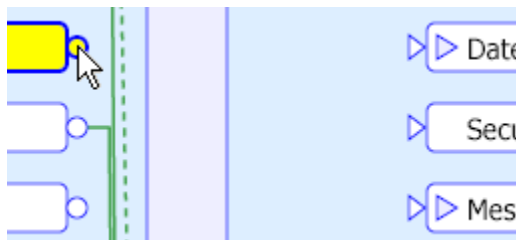
ソース・メッセージをコピーする assign アクションを作成するには：

1. ソース・メッセージの右にある円形のタブをドラッグします。マウス・ボタンを押さえます。
2. カーソルをターゲット・メッセージの左にある三角形のタブにそのボックスの色が変化するまでドラッグします。
3. 左マウス・ボタンを放します。

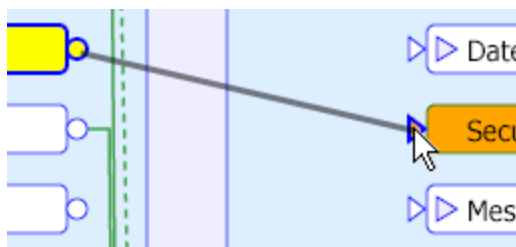
4.3 ソース・プロパティからターゲット・プロパティへの値のコピー

ソース・プロパティからターゲット・プロパティに値をコピーする assign アクションを作成するには:

1. ソース・プロパティからターゲット・プロパティに値をドラッグします。これを行うには、以下を実行します。
 - a. ソース・プロパティの右にある丸いタブをクリックします。マウス・ボタンを押さえます。画面には以下のように表示されます。



- b. カーソルをターゲット・プロパティの左にある三角形のタブにそのボックスの色が変化するまでドラッグします。画面には以下のように表示されます。



- c. 左マウス・ボタンを放します。画面には以下のように表示されます。



これで、ダイアグラムの下のテーブルに assign アクションの詳細が表示されます。

2. オプションで、[アクション] タブで詳細を編集します。

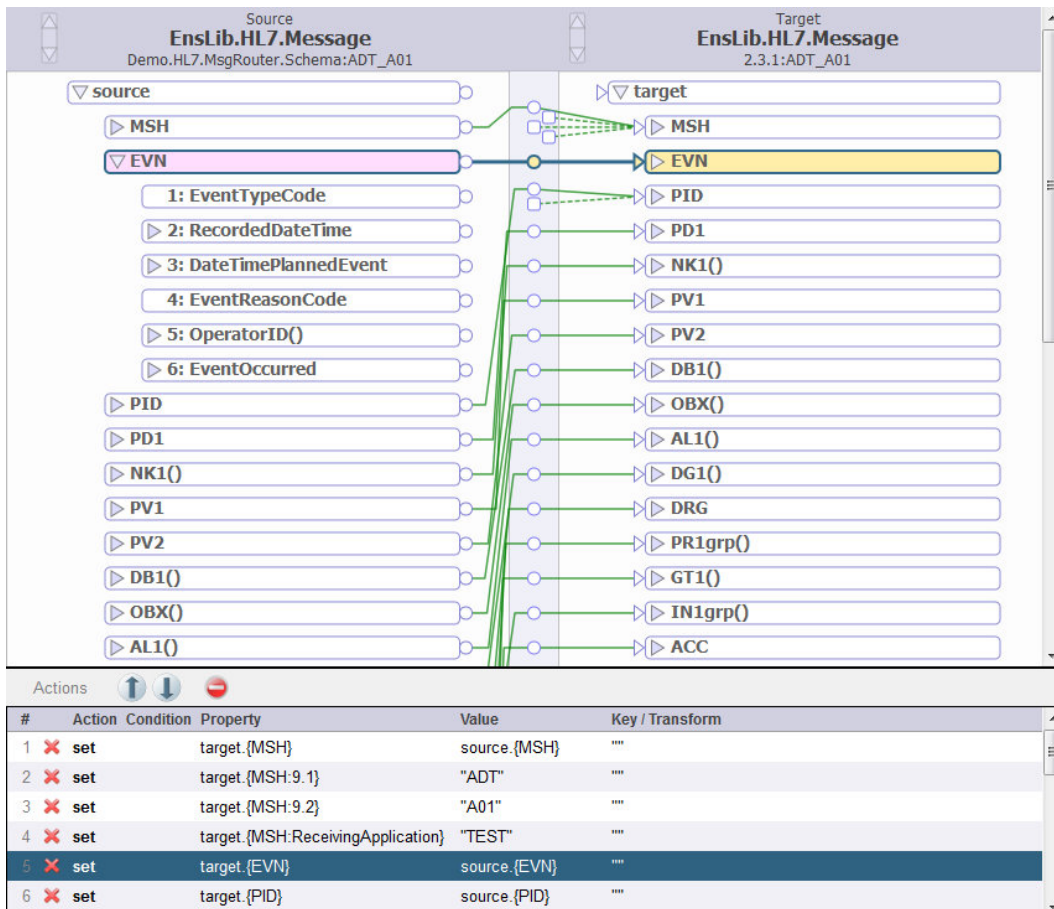
この assign アクションでは set が使用されています。

4.4 すべてのサブプロパティの値のコピー

ソースとターゲット内の親プロパティが同じであり、ソースとターゲットが同じタイプの場合は、すべてのサブプロパティの値を一度に割り当てることができます。その場合、親プロパティを展開してサブプロパティを表示する必要はありません。ソース側の親プロパティからターゲット側の親プロパティにカーソルをドラッグするだけです。

以下の DTL ダイアグラムでは、ソース側とターゲット側の EVN プロパティ間の 1 つのコネクタに次の EVN のサブプロパティのすべてが含まれます。

- ・ EventTypeCode
- ・ RecordedDateTime とそのすべてのサブプロパティ
- ・ DateTimePlannedEvent とそのすべてのサブプロパティ
- ・ EventReasonCode
- ・ OperatorID、そのすべての反復、およびそのすべてのサブプロパティ
- ・ EventOccurred とそのすべてのサブプロパティ



この assign アクションでは set が使用されています。

注釈 ソースとターゲットのタイプが異なる場合は、構造が並列であるように見える場合でも、この機能を使用してサブプロパティを割り当てることはできません。このような変換の場合は、構造の各リーフ・ノードを別々に割り当てて、for each アクションをプロセス反復に追加する必要があります。for each アクションの詳細は、“[For Each アクションの追加](#)”を参照してください。

4.5 ターゲット・プロパティへのリテラル値の割り当て

ターゲット・プロパティにリテラル値を割り当てるには：


1. ターゲット・プロパティを選択します。
2. [アクション追加] ドロップダウン・リストで [set] をクリックします。この操作の [アクション] タブが表示されます。
3. [値] フィールドにリテラルの数値または文字列値を入力します。
 - ・ 数値リテラルはただの数字です。例えば、42.3 などです。
 - ・ 文字列リテラルは二重引用符で囲まれた文字列です。例えば、"ABD" などです。

注釈 この文字列には XML 予約文字を含めることができません。仮想ドキュメントの場合は、この文字列に仮想ドキュメント形式で使用するセパレータ文字を含めることができません。詳細は、“[構文ルール](#)”の章を参照してください。

4. [保存] をクリックします。

4.6 ターゲット・プロパティの値に対する式の使用

前の節で説明したリテラル値は単純な式の種類です。ターゲット・プロパティの値としてもっと複雑な式を使用したい場合があります。これを実現するには、以下のどちらかを実行します。

- ・ 関数を使用した式を作成する場合は、[値] フィールドの横にある検索ボタン  をクリックします。これにより、以降の項で説明するデータ変換関数ウィザードが呼び出されます。
- ・ より複雑な式を作成する場合は、[値] フィールドに式を入力します。

前述した“[有効な式](#)”を参照してください。データ変換用に選択されたスクリプティング言語で書かれた式が有効なことを確認します。前述した“[変換詳細の指定](#)”を参照してください。

4.6.1 データ変換関数ウィザードの使用法

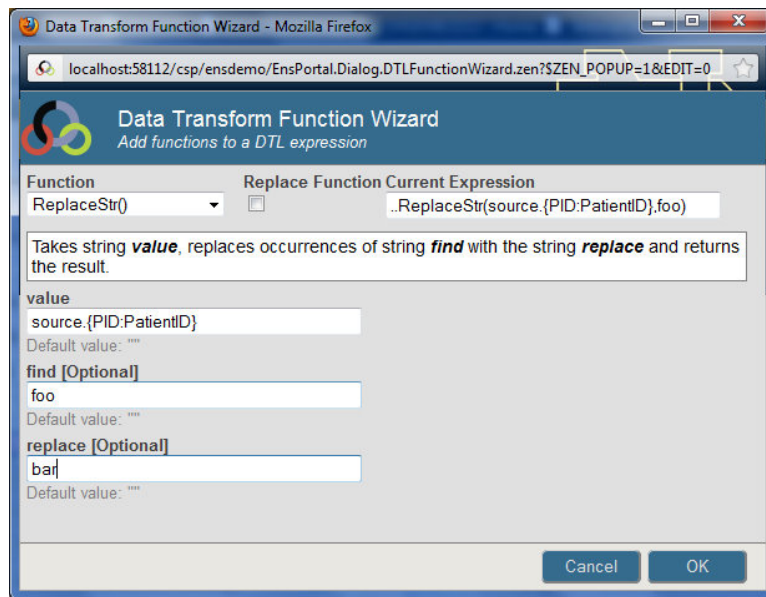
データ変換関数ウィザードを使用するには：

1. ドロップダウン・リストから [関数] を選択します。

必要に応じて、式を定義するための複数のフィールドが表示されます。

ドロップダウン・リストから [現在の関数を繰り返す] を選択した場合は、現在の関数のコピーがその関数自体のパラメータとして挿入されて、その関数の再帰呼び出しが作成されます。

2. 必要に応じてフィールドを編集します。手順は、ダイアログ内のコンテキスト依存ヘルプを参照してください。
3. [OK] をクリックして変更内容を保存し、ウィザードを終了します。



4.7 コレクション項目への値の割り当て

この節は、次の種類のコレクションに適用されます。

- ・ 標準プロダクション・メッセージ内のコレクション・プロパティ
- ・ XML 仮想ドキュメント内の繰り返しフィールド

コレクションから項目の値を変更するには：

1. ターゲット・リスト・プロパティまたはターゲット配列プロパティを選択します。
2. **[アクション追加]** ドロップダウン・リストで **[set]** をクリックします。この操作の **[アクション]** タブが表示されます。
3. **[プロパティ]** フィールドで、変更する項目を指定するようにかっこ内の値を編集します。

配列プロパティの場合は、配列項目のキーを使用します。リスト・プロパティの場合は、リスト項目のインデックスを使用します。仮想ドキュメント内の繰り返しフィールドの場合は、セグメントまたはフィールドのインデックスを使用します。

例えば、以下のように表示されていたとします。

```
target.MyArrayProp.(1)
```

代わりに以下を含むようにフィールドを編集します。

```
target.MyArrayProp("key2")
```

4. リテラル値またはその他の有効な式を含むように **[値]** を編集します。

前述した“[有効な式](#)”を参照してください。データ変換用に選択されたスクリプティング言語で書かれた式が有効なことを確認します。前述した“[変換詳細の指定](#)”を参照してください。

5. **[保存]** をクリックします。

以下に例を示します。

Action	set
Property	target.MyListProp.(3)
Property whose value will be set. Double-click diagram will place that property in this field.	
Value	"new value"
Value to assign to the property. Double-click diagram will place that property in this field.	

または、表示された値から最後の .(1) を削除するように [プロパティ] フィールドを編集します。その後で、[次の節](#)で説明するように、[キー] を使用して変更する項目を指定します。以下に例を示します。

Action	set
Property	target.MyListProp
Property whose value will be set. Double-click diagram will place that property in this field.	
Value	"new value"
Value to assign to the property. Double-click diagram will place that property in this field.	
Key	4
For collection properties, this string specifies the target of this assignment.	

4.8 リスト項目の挿入

この節は、標準プロダクション・メッセージ内のリスト・プロパティに適用されます（配列プロパティには適用されません）。このアクションは XML 仮想ドキュメントでも使用できます。“[プロダクション内での XML 仮想ドキュメントのルーティング](#)”を参照してください。

リストに項目を挿入するには：

1. ターゲット・リスト・プロパティまたはターゲット配列プロパティを選択します。
2. [アクション追加] ドロップダウン・リストで [insert] をクリックします。この操作の [アクション] タブが表示されます。
3. [プロパティ] フィールドで、表示された値から最後の .(1) を削除します。

例えば、以下のように表示されていたとします。

```
target.MyListProp.(1)
```

代わりに以下を含むようにフィールドを編集します。

```
target.MyListProp
```

4. リテラル値またはその他の有効な式を含むように [値] を編集します。

前述した“[有効な式](#)”を参照してください。データ変換用に選択されたスクリプティング言語で書かれた式が有効なことを確認します。前述した“[変換詳細の指定](#)”を参照してください。

5. **【キー】** で、新しい項目のインデックス位置を指定します。

以下に例を示します。

5

6. **【保存】** をクリックします。

以下に例を示します。

The screenshot shows a dialog box with the following fields and values:

- Action:** insert (dropdown menu)
- Property:** target.MyListProp
- Value:** "value to insert"
- Key:** 5

Below each field is a descriptive text:

- Property:** Property whose value will be set. Double-click diagram will place that property in this field.
- Value:** Value to assign to the property. Double-click diagram will place that property in this field.
- Key:** For collection properties, this string specifies the target of this assignment.

4.9 リスト項目の追加

この節は、標準プロダクション・メッセージ内のリスト・プロパティに適用されます（配列プロパティには適用されません）。このアクションは XML 仮想ドキュメントでも使用できます。“[プロダクション内での XML 仮想ドキュメントのルーティング](#)”を参照してください。

リストに項目を挿入するには：

1. ターゲット・リスト・プロパティまたはターゲット配列プロパティを選択します。
2. **【アクション追加】** ドロップダウン・リストで **[append]** をクリックします。この操作の **【アクション】** タブが表示されます。
3. **【プロパティ】** フィールドで、表示された値から最後の **.(1)** を削除します。

例えば、以下のように表示されていたとします。

```
target.MyListProp.(1)
```

代わりに以下を含むようにフィールドを編集します。

```
target.MyListProp
```

4. リテラル値またはその他の有効な式を含むように **【値】** を編集します。

前述した“[有効な式](#)”を参照してください。データ変換用に選択されたスクリプティング言語で書かれた式が有効なことを確認します。前述した“[変換詳細の指定](#)”を参照してください。

5. **【保存】** をクリックします。

以下に例を示します。

Action
append

Property
target.MyListProp
Property whose value will be set. Double-clicking the diagram will place that property in this field.

Value
"value to append"
Value to assign to the property. Double-clicking the diagram will place that property in this field.

Key
"
For collection properties, this string specifies the target of this assignment.

4.10 コレクション項目の削除

この節は、標準プロダクション・メッセージ内のコレクション・プロパティ (リストと配列) に適用されます。このアクションは XML 仮想ドキュメントでも使用できます。["プロダクション内での XML 仮想ドキュメントのルーティング"](#) を参照してください。

コレクションから項目を削除するには:

1. ターゲット・リスト・プロパティまたはターゲット配列プロパティを選択します。
2. **[アクション追加]** ドロップダウン・リストで **[remove]** をクリックします。この操作の **[アクション]** タブが表示されます。
3. **[プロパティ]** フィールドで、表示された値から最後の **.(1)** を削除します。

例えば、以下のように表示されていたとします。

```
target.MyArrayProp.(1)
```

代わりに以下を含むようにフィールドを編集します。

```
target.MyArrayProp
```

4. **[キー]** で、削除する項目を指定します。

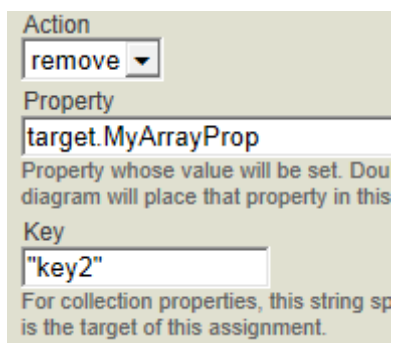
配列プロパティの場合は、配列項目のキーを使用します。リスト・プロパティの場合は、リスト項目のインデックスを使用します。仮想ドキュメント内の繰り返しフィールドの場合は、セグメントまたはフィールドのインデックスを使用します。

例えば、以下のように指定します。

```
"key2"
```

5. **[保存]** をクリックします。

以下に例を示します。



4.11 コレクション・プロパティの消去

この節は、標準プロダクション・メッセージ内のコレクション・プロパティ (リストと配列) に適用されます。このアクションは XML 仮想ドキュメントでも使用できます。[“プロダクション内での XML 仮想ドキュメントのルーティング”](#)を参照してください。

コレクションの内容を消去するには:

1. ターゲット・リスト・プロパティまたはターゲット配列プロパティを選択します。
2. **[アクション追加]** ドロップダウン・リストで **[clear]** をクリックします。この操作の **[アクション]** タブが表示されます。
3. **[プロパティ]** フィールドで、表示された値から最後の .(1) を削除します。

例えば、以下のように表示されていたとします。

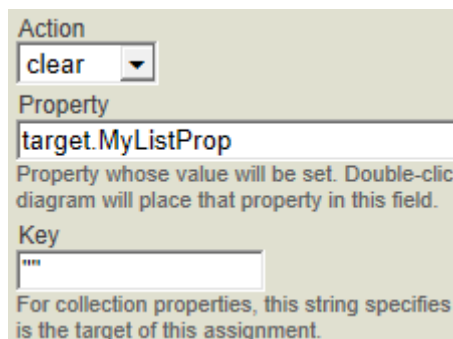
```
target.MyArrayProp.(1)
```

代わりに以下を含むようにフィールドを編集します。

```
target.MyArrayProp
```

4. **[保存]** をクリックします。

以下に例を示します。



5

その他のアクションの追加

この章では、DTL 変換にその他の種類のアクションを追加する方法について詳しく説明します。

アクションのグループを操作する方法の詳細は、“[アクションのグループの操作](#)”を参照してください。

assign アクションの追加方法は、[前の章](#)を参照してください。

5.1 If アクションの追加

if アクションは、指定された式の値に応じて、その他のアクションを条件付きで実行します。InterSystems IRIS® では、それぞれの if アクションが [DTL ダイアグラム](#)内のコネクタ線で表現されます。

if アクションを追加するには：

1. 条件がソース・プロパティの値に依存する場合は、そのプロパティをクリックします。
2. **[アクション追加]** ドロップダウン・リストから **[if]** を選択します。

[アクション] タブでは、**[条件]** フィールドに、選択されたソース・プロパティの名前が自動的に入力されます。

ダイアグラムの下領域に 3 つの新しい行が表示されます。次のように、これらの行のラベルが **[アクション]** 列に表示されます。


- ・ if – この行は、条件が真の場合に実行するアクションの始まりを表します。
- ・ else – この行は、条件が偽の場合に実行するアクションの始まりを表します。
- ・ endf – この行は、if アクションの終わりを表します。

3. 真または偽に評価される式を含めるように **[条件]** フィールドを編集します。

以下に例を示します。

```
source.ABC = "XYZ"
```

注：

- ・ 関数を使用した式を作成する場合は、**[値]** フィールドの横にある検索ボタン  をクリックします。これにより、[前述したデータ変換関数ウィザード](#)が呼び出されます。
- ・ より複雑な式を作成する場合は、**[値]** フィールドに式を入力します。前述した“[有効な式](#)”を参照してください。データ変換用に選択されたスクリプティング言語で書かれた式が有効なことを確認します。前述した“[変換詳細の指定](#)”を参照してください。

4. 条件が真の場合に実行するアクションを追加するには:

- a. if 行をクリックします。
- b. **[アクション追加]** ドロップダウン・リストから項目を選択します。
- c. 必要に応じて、**[アクション]** タブで値を編集します。
- d. 必要に応じて繰り返します。

assign アクション、if アクション、および for each アクションを含めることができます。

5. 条件が偽の場合に実行するアクションを追加するには:

- a. else 行をクリックします。
- b. 前の項目で説明したように先に進みます。

これで、DTL ダイアグラムの下のブロック内に詳細が表示されます。以下に例を示します。

6	×	if	source.{PID:PatientIdentifierList(k1).id...	
7	×	set	target.{PID:PatientIdentifierList(k1).as...	"AUSHIC" ""
8	×	set	target.{PID:PatientIdentifierList(k1).id...	"MC" ""
9		else		
10		endif		

注釈 if 分岐または else 分岐用のアクションを指定する必要はありません。どの分岐にもアクションが含まれていない場合は、if アクションが無視されます。

5.2 For Each アクションの追加

for each アクションでは、繰り返し実行される一連のアクションを、以下のいずれかの各メンバに対して 1 回定義できます。

- ・ コレクション・プロパティ (標準メッセージ)
- ・ 繰り返しプロパティ (仮想ドキュメント)
- ・ ドキュメント内の一連のサブドキュメント (仮想ドキュメント)

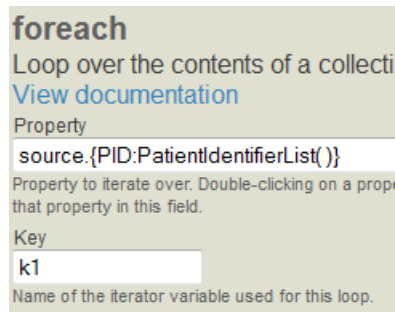
InterSystems IRIS では、それぞれの for each アクションが **DTL ダイアグラム** 内のコネクタ線で表現されます。

for each ループ内に break アクションを追加することにより、いつでもループを終了することができます。

for each アクションを追加するには:

1. ソース・メッセージ内のコレクション・プロパティまたは繰り返しプロパティを選択します。
2. **[アクション追加]** ドロップダウン・リストから **[for each]** を選択します。

以下に示すように、**[アクション]** タブの **[プロパティ]** フィールドには自動的に選択されたソース・プロパティの名前が含まれ、**[キー]** フィールドには自動的に k1 が含まれます。



for each アクションの場合は、[キー] フィールドでカウンタ変数の名前を指定します。

[プロパティ] フィールドでは、反復子キーを括弧内に含めることはできません。例として、以下は正しい指定です。

```
source.{PID:PatientIdentifierList( )}
```

for each は PatientIdentifierList 繰り返しフィールド (最初のフィールド (1 番) から始まって最後のフィールドで終わる) を通して繰り返します。

3. [アクション] タブの [Unload] チェック・ボックスは、オープン・オブジェクトまたはセグメントのアンロードのためのコードを生成するかどうかを制御します。

for each アクションに対して [Unload] にチェックが付いている場合、Transform メソッドで各ループの最後のプロパティ・コレクションに対してオープン・オブジェクトまたはセグメントのアンロード/アンスウィズルを試みるコードが生成されます。保存されていない仮想ドキュメント・セグメントが保存され、完成されます。プロパティがソース・オブジェクトの場合、ソース・オブジェクトは通常既に保存されています。

それでもなお、ターゲットのコレクションのオブジェクトまたはセグメントをアンロードするため、手動でアクションを追加する必要がある場合があります。方針の詳細は、“[ターゲット・コレクションのアンロード](#)” を参照してください。

for each プロパティ・コレクションのアンロードは不要の可能性があります。例えば、HL7 の場合、CopyValues を使用して生成されたコードはソース・セグメントをインスタンス化しません。

4. for each ブロックにアクションを追加するには、for each アクションをクリックしてから、該当するアクションを追加します。

これで、DTL ダイアグラムの下のブロック内に詳細が表示されます。以下に例 (一部) を示します。

#	Action	Condition	Property	Value	Key / Transform
1	✗ for each		source.{PID:PatientIdentifierList()}		k1
2	✗ if	source.{PID:PatientIdentifierList(k1).id...			
3	✗ set		target.{PID:PatientIdentifierList(k1).id...	"MR"	—
4	else				
5	endif				
6	✗ if	source.{PID:PatientIdentifierList(k1).id...			

<foreach> がメッセージ内のコレクション・プロパティに適用される場合、一連のアクティビティは繰り返し実行されますが、コレクション・プロパティ内に存在するすべての要素に対して 1 回実行されます。要素が NULL の場合、このシーケンスは実行されません。このシーケンスは、要素に空の値がある場合に (つまり、セパレータはあるがその間に値がない場合に) 実行されます。しかし、NULL 値に対しては実行されません。つまり、フィールドが指定される前にメッセージが終了します。

5.2.1 For Each アクションのショートカット

仮想ドキュメントを操作している場合は、ドキュメント構造内の繰り返しフィールドのすべてのインスタンスを通して繰り返すショートカット表記が提供されます。これは、実際には、繰り返しフィールドを処理するためのネストされた複数の for each ループをセットアップする必要がないことを意味します。代わりに、中かっこ {} 構文内の空の丸かっこを含む仮想プロパティ・パスを使用して単一の assign アクションを作成します。詳細は、“プロダクション内での仮想ドキュメントの使用法”の“中かっこ {} の構文”を参照してください。

注釈 ソースとターゲットのタイプが異なる場合は、このショートカットを for each アクションに対して使用することはできません。このような場合は、明示的な for each アクションを使用してください。

5.2.2 ターゲット・コレクションのアンロード

[Unload] オプションを使用するとソース・コレクションからオブジェクトが自動的に削除されますが、ターゲット・コレクションからオブジェクトを削除するには、for each アクションの末尾にカスタム・コードを追加する必要があります。ターゲットが複雑なレコードである場合の簡単な例では、以下のコードを使用して現在のターゲット・レコードを保存してから、レコードをアンロードできます。

```
Do target.Record16.GetAt(k1).%Save(0)
Do target.Record16.%UnSwizzleAt(k1)
```

他のシナリオでは、ターゲットがアンロードされない問題を回避するために、ターゲットをまったくロードしないことをお勧めします。例えば、あるオブジェクトに、多数の子を持つ親子プロパティが指定されているとします。for each アクション内で、サブ変換を `propSetObjectId(parentId)` と組み合わせています。prop は、プロパティの名前です。

この例では、ターゲットはバッチ・オブジェクト、ターゲット・クラスは `Demo.RecordMapBatch.Map.TrainDataOut.BatchOut`、レコード・クラスは `Demo.RecordMapBatch.Transform.Optimized.Record` です。

for each ループの前に、空のターゲットを作成してその ID をプロパティ `BatchOutID` に割り当てる必要があります。

```
<assign value='target.%Save()' property='tSC' action='set' />
<assign value='target.%Id()' property='BatchOutID' action='set' />
<assign value='target' property='' action='set' />
```

その後、for each ループ内で、ターゲットをインスタンス化することなくターゲットに直接作用するコードを使用できます。以下に例を示します。

```
<assign value=''' property='record' action='set' />
<subtransform class='Demo.RecordMapBatch.Transform.Optimized.Record' targetObj='record'
sourceObj='source.Records.(k1)' />

<comment>
<annotation>Assign record to target directly. </annotation>
</comment>
<assign value='record.%ParentBatchSetObjectId(BatchOutID)' property='tSC' action='set' />
<assign value='record.%Save()' property='tSC' action='set' />
```

続いて、DTL が終了する前に、変数 `target` の設定を、予期される DTL の結果に戻します。以下に例を示します。

```
<assign value='##class(Demo.RecordMapBatch.Map.TrainDataOut.BatchOut).%OpenId(BatchOutID)'
property='target' action='set' />
```

5.2.3 ラージ・メッセージでの <STORE> エラーの回避

メッセージまたはオブジェクト・コレクションでセグメントのループ処理を行う際には、セグメントがメモリに書き込まれます。これらのオブジェクトによって現在のプロセスに割り当てられているすべてのメモリが消費されると、予期しないエラーが発生することがあります。このようなエラーを回避するには、管理ポータルで [Unload] オプションを使用します。ターゲッ

ト・コレクションのオブジェクトを削除する場合の方針については、“[ターゲット・コレクションのアンロード](#)”を参照してください。

別の方針として、for each ループで多数のセグメントを処理する場合は、ループの最後のステップとしてソースとターゲットの両方に対して commitSegmentByPath() メソッドを呼び出すことができます。同様に、オブジェクト・コレクションの場合は、%UnSwizzleAt() メソッドを使用します。

メソッド commitCollectionOpenSegments() は、runtimePath をループして指定されたコレクション・パス内のオープン・セグメントを検索し、各オープン・セグメントの commitSegmentByPath() を呼び出します。このメソッドは、クラス EnsLib.EDI.X12.Document、EnsLib.EDI.ASTM.Document、EnsLib.EDI.EDIFACT.Document、および EnsLib.HL7.Message から利用できます。

コードを変更できない場合の一時的な回避策としては、各プロセスに割り当てられているメモリの量を増やします。この変更を行うには、管理ポータルの **[メモリ詳細設定]** ページで bbsiz パラメータを設定します。このアクションはシステムを再起動する必要があるため、これを実行する前に、システム管理者に相談してください。

5.3 Subtransform アクションの追加

subtransform は通常、for each ループ内で別の変換（一般変換）を呼び出します。EDI 形式は複数のメッセージ・タイプで使用するセグメント・セットに基づいていることが多いため、サブ変換は、特に、仮想ドキュメントに有効です。別の変換内で変換を再利用できるということは、コード変換を複製しなくても必要に応じて呼び出すことが可能なセグメント変換の再利用可能ライブラリを作成できることを意味します。

[DTL ダイアグラム](#)には subtransform アクションが表示されません。

subtransform アクションを追加するには：

1. **[アクション追加]** ドロップダウン・リストから **[subtransform]** を選択します。
2. **[アクション]** タブで、以下の情報を指定します。
 - ・ **[変換クラス]** – 使用するデータ変換クラスを指定します。これは、DTL 変換にすることも、カスタム変換にすることもできます。カスタム変換の詳細は、“[プロダクションの開発](#)”の“[カスタム変換の定義](#)”を参照してください。該当するクラスを入力する必要があります。
 - ・ **[ソースプロパティ]** – 変換するプロパティを指定します。これは、オブジェクト・プロパティにすることも、仮想ドキュメント・プロパティ・パスにすることもできます。通常は、変換に使用されるソース・メッセージのプロパティです。該当するソース・プロパティを入力する必要があります。
 - ・ **[ターゲットプロパティ]** – 変換された値が書き込まれるプロパティを指定します。これは、オブジェクト・プロパティにすることも、仮想ドキュメント・プロパティ・パスにすることもできます。通常は、変換に使用されるターゲット・メッセージのプロパティです。該当するターゲット・プロパティを入力する必要があります。
 - ・ **[予備のプロパティ]** – 必要に応じて、サブ変換に渡される値を指定します。サブ変換は、この値に aux 変数としてアクセスします。
 - ・ **[無効]** – 必要に応じて、サブ変換を無効にするように指定します。
 - ・ **[説明]** – 必要に応じて、サブ変換の説明文を入力します。

注釈 **[作成]** が new または copy に設定されている subtransform の場合は、既存のターゲット・オブジェクトが含まれている必要がありません。

5.4 Trace アクションの追加

trace アクションは、診断に役立つトレース・メッセージを生成します。[\[トレース・イベントを記録\]](#) 設定が親ビジネス・ホストに対して有効になっている場合は、このメッセージがイベント・ログに書き込まれます。[\[フォアグラウンド\]](#) 設定が親ビジネス・ホストに対して有効になっている場合は、トレース・メッセージがターミナル・ウィンドウにも出力されます。

[DTL ダイアグラム](#)には trace アクションが表示されません。

trace アクションを追加するには:

1. **[アクション追加]** ドロップダウン・リストから **[trace]** を選択します。
2. **[アクション]** タブで、以下を指定します。
 - ・ **[値]** – リテラル値またはその他の有効な式を指定します。
前述した“[有効な式](#)”を参照してください。データ変換用に選択されたスクリプティング言語で書かれた式が有効なことを確認します。前述した“[変換詳細の指定](#)”を参照してください。
 - ・ **[説明]** – 任意で説明を指定します。

trace アクションは、ユーザ優先度を持つトレース・メッセージを生成します。結果は、ObjectScript 内の \$\$\$TRACE マクロを使用した場合と同じです。

5.5 Code アクションの追加

code アクションを使用すれば、DTL データ変換内で 1 行または複数行のユーザ作成コードを実行できます。このオプションを使用すれば、DTL 要素では表現しづらい特殊なタスクを実行できます。[DTL ダイアグラム](#)には code アクションが表示されません。

code アクションを追加するには:

1. **[アクション追加]** ドロップダウン・リストから **[code]** を選択します。
2. **[アクション]** タブで、以下を指定します。
 - ・ **[コード]** – 変換用に指定されたスクリプティング言語で 1 行または複数行のコードを指定します。このコード内の式に関するルールは、“[構文ルール](#)”を参照してください。
InterSystems IRIS は、自動的に、コードを CDATA ブロックで囲みます。これは、アポストロフィ (') やアンパサンド (&) などの特殊な XML 文字でエスケープする必要がないことを意味します。
後述する注意も参照してください。
 - ・ **[説明]** – 任意で説明を指定します。

Tip デバッグを容易にするためのカスタム・コードを作成するには、ターミナルで実行できるようにクラス・メソッドまたはルーチン内にコードを含めます。ターミナルでコードをデバッグします。その後で、DTL の code アクション内部からそのメソッドまたはルーチンを呼び出します。

5.5.1 DTL 内でカスタム・コードを使用する場合のガイドライン

データ変換の実行を中断して再開できるようにするには、code アクションを使用するときに以下のガイドラインに従う必要があります。

- ・ 実行時間は短くします。カスタム・コードがデータ変換の一般処理を妨げないようにしてください。
- ・ システム・リソースを割り当てる（ロックの取得やデバイスのオープンなど）場合は、必ず同じ code アクション内でそのリソースを解放してください。
- ・ code アクションでトランザクションを開始する場合は、考えられるすべてのシナリオにおいて同じアクションでトランザクションが終了することを確認します。そうしなければ、トランザクションを閉じることができなくなる可能性があります。これにより他の処理が阻止されたり、重大なダウンタイムが発生することがあります。

5.6 SQL アクションの追加

SQL アクションを使用すれば、DTL 変換内部から SQL SELECT 文を実行できます。[DTL ダイアグラム](#)には sql アクションが表示されません。

sql アクションを追加するには：

1. **[アクション追加]** ドロップダウン・リストから **[sql]** を選択します。
2. **[アクション]** タブで、以下を指定します。
 - ・ **SQL** — 有効な SQL SELECT 文を指定します。
InterSystems IRIS は、自動的に、SQL を CDATA ブロックで囲みます。これは、アポストロフィ (') やアンパサンド (&) などの特殊な XML 文字でエスケープする必要がないことを意味します。
後述する注意も参照してください。
 - ・ **[説明]** — 任意で説明を指定します。

5.6.1 DTL 内で SQL を使用する場合のガイドライン

必ず、以下のガイドラインを使用してください。

- ・ 次のように、必ず、SQL スキーマ名とテーブル名の両方を含むテーブルの完全修飾名を使用します。

```
MyApp.PatientTable
```

上記の例の MyApp は SQL スキーマ名、PatientTable はテーブル名です。

- ・ FROM 節内に列挙するテーブルは、ローカルの InterSystems IRIS データベース内に保存されているか、SQL ゲートウェイを通して外部リレーショナル・データベースにリンクされている必要があります。
- ・ SQL クエリの INTO 節と WHERE 節から、ソースまたはターゲット・オブジェクトのプロパティを参照できます。これを実現するには、プロパティ名の前にコロンの (:) を付けます。以下に例を示します。

```
SELECT Name INTO :target.Name FROM MainFrame.EmployeeRecord WHERE SSN = :source.SSN AND City = :source.Home.City
```

- ・ 使用されるのは、クエリで返された最初の行のみです。WHERE 節では、必要な行を正確に指定してください。

5.7 Switch アクションの追加

switch アクションには、1 つまたは複数の一連の case アクションと 1 つの default アクションが含まれています。switch アクションが実行されると、各 case 条件の評価が開始されます。式が true として評価されると、対応する case ブロックの内容が実行されます。それ以外の場合は、次の case アクションの式が評価されます。いずれかの case アクションが実行されるとすぐに、変換の実行パスは、他の条件を評価することなく、switch ブロックから離れます。いずれの case 条件も true でない場合は、default アクションの内容が実行された後、switch ブロックから制御が離れます。

5.8 Case アクションの追加

条件が一致した場合にアクションのブロックを実行するには、switch ブロック内で case アクションを使用します。case 条件が満たされ、アクションのブロックが実行されると、変換の実行パスは、他の条件を評価することなく、switch ブロックから離れます。

case アクションを追加するには:

1. ダイアグラムの下のリストで switch アクションを選択します。
2. **[アクション追加]** ドロップダウン・リストから **[case]** を選択します。
3. **[アクション]** タブで、条件を追加します。虫めがねアイコンをクリックして、関数を条件の一部として追加することができます。
4. ダイアグラムの下のリストで case アクションを選択した状態で、**[アクション追加]** ドロップダウンを使用して、条件が true として評価された場合に実行するアクションを追加します。

5.9 Default アクションの追加

[アクション追加] ドロップダウン・リストを使用して、default ブロックを追加することはできません。switch アクションを追加すると、default アクションが switch ブロックに自動的に追加されます。switch ブロック内のいずれの case 条件も満たされない場合、default ブロックに含まれているアクションが実行されます。いずれの case 条件も満たされない場合に何も実行されないようにするには、単に default ブロックを空のままにします。

5.10 Break アクションの追加

break アクションが実行されるとすぐに for each ループを終了するには、break アクションをループに追加します。break アクションが実行された後、データ変換は引き続き、for each ループのすぐ後のアクションを処理します。

break アクションを for each ループの外に追加した場合、データ変換は、break アクションが実行されるとすぐに終了します。

5.11 Comment アクションの追加

データ変換のアクションにアノテーションを付けるために、アクションのリストに表示されるコメントを追加できます。[アクション追加]→[コメント]を選択した後、[アクション] タブの [説明] テキスト・ボックスにコメントを入力します。

6

データ変換のテスト

データ変換クラスをコンパイルしたら、それをテストできます (テストすべきです)。この章では、その方法について説明します。

注釈 この章は、DTL 変換とカスタム変換の両方に適用されます。

6.1 変換テスト・ページの使用方法

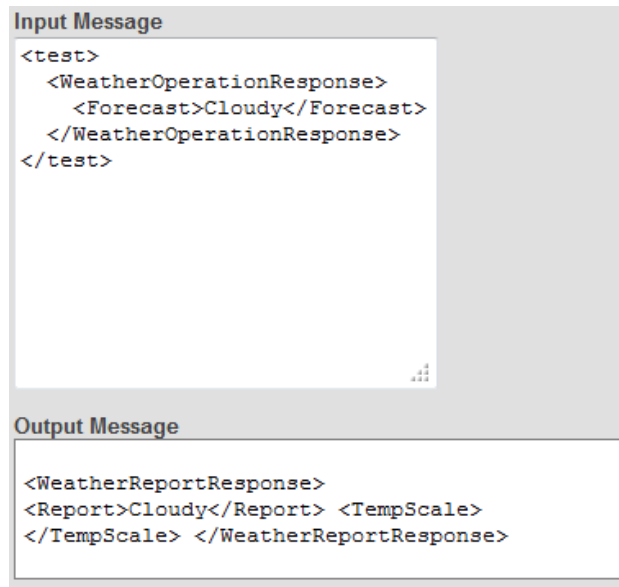
管理ポータルで、**変換テスト**・ウィザードを使用できます。このウィザードには、管理ポータルの以下の場所からアクセスできます。

- ・ データ変換ビルダの **[ツール]** タブで **[テスト]** をクリックします。
- ・ 変換を選択して、**[データ変換リスト]** ページで **[テスト]** をクリックします。

初期状態では **[出力メッセージ]** ウィンドウは空で、**[入力メッセージ]** ウィンドウにはソース・メッセージに適した形式のテキスト・スケルトンが入力されています。テストするには、以下の手順を実行します。

1. DTL コードが aux、コンテキスト、またはプロセス・システム・オブジェクトのプロパティを参照する場合、これらのプロパティの値を入力して、これらのオブジェクトがインスタンス化された状態でデータ変換が呼び出された場合と同様に結果を表示します。値を入力するテーブルが表示されるのは、DTL が aux、プロセス、またはコンテキスト・システム・オブジェクトの内部プロパティを参照する場合のみです。
2. 適切なデータが含まれるように、**[入力メッセージ]** を編集します。この入力ボックスに表示される内容と入力する内容は、次のようにソースのタイプとクラスに応じて異なります。
 - ・ EDI メッセージの場合は、このウィンドウには生のテキストが表示されます。したがって、いくつかの保存されたテキスト・ファイルを用意し、これらのファイルから **[入力メッセージ]** ボックスにテキストをコピー・アンド・ペーストできるようにします。
 - ・ 通常のプロダクション・メッセージの場合は、このウィンドウには、メッセージ・オブジェクト内の各プロパティのエントリを持つ XML スケルトンが表示されます。したがって、各プロパティの値を入力します。
 - ・ レコード・マップ、複雑なレコード・マップ、およびバッチ・レコード・マップについては、未加工のテキストまたは XML を入力できます。
3. **[テスト]** をクリックします。
4. **[出力メッセージ]** ボックスで結果を確認します。

以下に例を示します。



6.2 プログラムによる変換のテスト

変換をプログラムを使ってテストするには、ターミナルで以下を実行します（またはこれらのステップを含むルーチンまたはクラス・メソッドを作成します）。

1. ソース・メッセージ・クラスのインスタンスを作成します。
2. そのインスタンスのプロパティを設定します。
3. 変換クラスの Transform() クラス・メソッドを呼び出します。このメソッドには、以下のシグニチャがあります。

```
classmethod Transform(source As %RegisteredObject, ByRef target As %RegisteredObject) as %Status
```

説明：

- ・ source はソース・メッセージです。
 - ・ target は変換によって作成されたターゲット・メッセージです。
4. ターゲット・メッセージを検査して、期待どおりに変換されているかどうかを確認します。XML 形式の両方のメッセージを簡単に検査するには、次の手順を実行します。
 - a. %XML.Writer のインスタンスを作成します。
 - b. オプションで、そのインスタンスの Indent プロパティを 1 に設定します。
これにより、出力に改行が追加されます。
 - c. ソース・メッセージを引数として渡す Writer インスタンスの RootObject() メソッドを呼び出します。
 - d. Writer インスタンスを強制終了します。
 - e. ターゲット・メッセージで繰り返します。

以下に例を示します。

ObjectScript

```
//create an instance of the source message
set source=##class(DTLTest.Message).CreateOne()
set writer=##class(%XML.Writer).%New()
set writer.Indent=1 do writer.RootObject(source)
write !!
set sc=##class(DTLTest.Xform1).Transform(source,.target)
if $$$ISERR(sc)
    {do $system.Status.DisplayError(sc)}
set writer=##class(%XML.Writer).%New()
set writer.Indent=1
do writer.RootObject(target)
```

