HL7 の生産性ツール

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

HL7 の生産性ツール
InterSystems　Version 2023.1　2024-01-02

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

メッセージ・アナライザ

メッセージ・アナライザは、以下を実行できるコマンド行ユーティリティです。

- HL7 メッセージをスキャンして、指定されたドキュメント構造と比較し、メッセージに一致する新しいドキュメント構造を派生させる。通常、派生されたドキュメント構造には、スキャンされたメッセージで見つかった Z セグメントが含まれます。
- 指定されたドキュメント構造に照らしてメッセージを検証し、オプションでカスタム・スキーマのセグメント、データ構造、およびコード・テーブルを更新し、失敗メッセージに対応できるようにする。

メッセージ・アナライザでは以下の用語を使用します。

用語	説明
ドキュメント構造	ADT_01 などの HL7 メッセージ構造。管理ポータルでは、スキーマのドキュメント構造は [DocType 構造] タブに表示されます。
HL7 構成	管理ポータルで定義されているスキーマ、ドキュメント構造、セグメント、データ構造、およびコード・テーブルの総称です。
ライブラリ・スキーマ	管理ポータルで利用可能なベース・スキーマ (例えば 2.7.1)。カスタム・スキーマは、これらのライブラリ・スキーマの 1 つに基づきます。

1.1 メッセージ・アナライザの実行

メッセージ・アナライザを実行するには、以下の手順に従います。

1. InterSystems ターミナルを開きます。
2. HL7 メッセージをルーティングする HL7 相互運用プロダクションが含まれているか、これから含まれる予定のネームスペースに移動します。これは、カスタム・スキーマが保存されるネームスペースです。例えば、以下のように入力します。

   ```
   set $namespace="MyRoutingNamespace"
   ```

3. 以下のように入力します。

   ```
   Do ##class(EnsLib.InteropTools.HL7.MessageAnalyzer).Interactive()
   ```

メッセージ・アナライザにより、指定できるオプションを示す一連のプロンプトが表示されます。任意のプロンプトで ^ と入力すると、コンテキストに応じて、前の入力に戻るかプログラムを終了します。メニューの中で適用されるオプションが 1

つだけの場合、メニューは表示されず、そのオプションが自動的に選択されます。プロンプトで *<value>* 構文を使用している場合、Enter キーを押すと、value がプロンプトへの回答として使用されます。

1.1.1 初期設定

メッセージ・アナライザを初めて実行する場合、ワークスペース・フォルダ、および HL7 スキーマとドキュメント構造の情報を指定するよう求めるプロンプトが表示されます。

メッセージ・アナライザを次回実行するときには、初期設定はスキップされ、メイン・メニューで起動します。ワークスペース・フォルダを切り替える必要がある場合は、別のメッセージをロードするか、別のスキーマ/ドキュメント構造を使用して、メイン・メニューから [ ] オプションを選択します。

1.1.1.1 ワークスペース・フォルダ

メッセージ・アナライザを初めて実行すると、ワークスペース・フォルダを指定するよう求めるプロンプトが表示されます。

```
Workspace folder:
```

現在存在しないファイル・システム・フォルダ (ディレクトリ) の名前を入力します。このフォルダはスクラッチ・ワークスペースとして使用されます。初期設定時には、ファイルが上書きされるのを避けるため、既存のフォルダを入力すると拒否されます。後で設定を再実行する場合は、既存のワークスペースを使用することも、新しいワークスペースを作成することもできます。

1.1.1.2 ソース・フォルダ

以下のプロンプトが表示され、ソース・ファイルまたはソース・フォルダを指定するよう求められます。

```
Source file or folder:
```

HL7 メッセージ・ファイル、または HL7 メッセージ・ファイルが含まれるフォルダを入力します。HL7 メッセージ・ファイルは、各行が 1 つのセグメントを表すテキスト・ファイルです。1 つのメッセージ・ファイルに任意の数のメッセージを含めることができます。新しいメッセージが始まる位置は、MSH セグメントを使用して検出されます。ファイルにメッセージのバッチを含めることもできます。

1.1.1.3 スキーマ

以下のプロンプトが表示され、スキーマを指定するよう求められます。

```
Schema:
```

エントリは以下のいずれかである必要があります。

- 既存のカスタム・スキーマ。
- 既存のライブラリ・スキーマ (例えば、2.4)。このスキーマを直接使用することはできないため、ライブラリ・スキーマに基づく新しいカスタム・スキーマの名前を指定するよう求めるプロンプトが表示されます。
- 新しいカスタム・スキーマ。カスタム・バージョンの基にする既存のライブラリ・スキーマを指定するよう求めるプロンプトが表示されます。

メッセージ・アナライザは、新しいドキュメント構造を派生させ、入力メッセージに基づいてこのカスタム・スキーマに変更を加えます。また、セグメント、データ構造、およびコード・テーブルを検証および変更し、失敗メッセージに対応できるようにする場合は、ライブラリ・スキーマではなくカスタム・スキーマも更新します。

1.1.1.4 ドキュメント構造

以下のプロンプトが表示され、ドキュメント構造を指定するよう求められます。

```
Existing document structure name, or a new one to be copied from another schema:
```

カスタム・スキーマに存在するドキュメント構造の名前 (例えば、ADT_A01)、または他のスキーマに存在するドキュメント構造の名前を入力します。カスタム・スキーマ内にあるドキュメント構造のリストから選択するには、ドキュメント構造の最初の文字と、それに続いて ? を入力します。例えば、A? と入力すると、ADT_A01 や ADT_A02 が表示されます。新しいドキュメント構造の名前を入力した場合、そのドキュメント構造をカスタム・スキーマにコピーするため、ドキュメント構造が存在するライブラリ・スキーマを指定するよう求めるプロンプトが表示されます。

メッセージ・アナライザは、新しいドキュメント構造を派生するときに、入力 HL7 メッセージの構造を、指定されたドキュメント構造と比較します。また、メッセージを検証するときに、指定されたドキュメント構造のセグメント、データ構造、およびコード・テーブルをメッセージ内のものと比較します。

管理ポータルでは、ドキュメント構造はスキーマの [DocType 構造] タブに表示されます。

1.1.2 メイン・メニュー

メッセージ・アナライザの初期設定を少なくとも 1 回完了すると、メイン・メニューが表示されます。

```
[1] Derive new document structures
[2] Validate messages, optionally fixing the configuration
[3] View history of fixes to the configuration (validation fixes only)
[4] Setup Workspace
```

[新規ドキュメント構造の派生] オプションを選択すると、入力メッセージが、設定中に指定されたドキュメント構造と照合され、メッセージの構造に基づいて新しいドキュメント構造が派生されます。

[メッセージを検証し、オプションで構成を修復] オプションを選択すると、入力メッセージが、指定されたドキュメント構造に照らして比較され、失敗メッセージに対応できるようにして検証に合格するためにスキーマの変更が提案されます。

[構成の修正履歴を表示] オプションを選択すると、カスタム・スキーマに加えられた変更のリストが表示されます。

[ワークスペースの設定] オプションを選択すると、初期設定プロセスが再実行され、新しいワークスペース、カスタム・スキーマ、またはドキュメント構造を指定できます。

1.1.3 新しいドキュメント構造の派生

[ ] オプションを選択すると、入力 HL7 メッセージが、設定中に指定されたドキュメント構造に対して照合され、メッセージの構造に基づいて新しいドキュメント構造が派生されます。例えば、派生されたドキュメント構造には、メッセージで使用されている Z セグメントが含まれる場合があります。

メッセージの分析プロセスには時間がかかる可能性があるため、[ ] オプションでは、スキャンするメッセージの割合を入力するよう求めるプロンプトが表示されます。

```
Enter a number between 1 and 100, or ^ to quit.
```

結果は以下のようになります。

```
Begin document structure derivation run
Loading document structure definition
Match messages to document structure: ORU_R01
Derive new document structure
Match messages to derived document structure
Finished.
Summary Report for workspace /Users/MSmith/wf2temp
Schema is MySchema24
 ALL: 19 matched, 53 unmatched, derived 1 new document structures
```

新しいドキュメント構造が派生されたら、新しいドキュメント構造のセグメントが表示されます。この場合、カスタム・スキーマのドキュメント構造を更新するかどうかを尋ねられます。

```
Updating MySchema24:ORU_R01 Would you like to update the document structure in the database? (Y/N):
```

1.1.4 メッセージの検証と構成の修正

メッセージ・アナライザの [ ] オプションは、カスタム・スキーマのセグメント、データ構造、およびコード・テーブルに照らして HL7 メッセージを検証します。カスタム・スキーマにこれらが存在しない場合は、カスタム・スキーマの基になっているライブラリ・スキーマに照らして検証します。検証が完了すると、メッセージ・アナライザは、入力メッセージのセグメント、データ構造、およびコード・テーブルに基づいてカスタム・スキーマを変更するよう提案し、メッセージがスキーマに対する検証に合格するようにします。

```
Workflow/2 - Validation
 [1] Select Field Validation [ ]
 [2] Select Component Validation [ ]
 [3] Select Code Table Validation [ ]
 [4] Start Validation
Enter one of the above options, ^ to go back a level:
```

メッセージ・アナライザでそのタイプの検証を実行するよう指定するには、最初の 3 つのエントリの 1 つを選択します。メニュー行が "[ ]" で終わる場合、その項目は検証で使用されません。メニュー行が "[X]" で終わる場合、その項目は検証で使用されます。エントリを 1 つ以上選択したら、4 を入力して検証を開始します。

フィールドの検証では、メッセージ内のセグメントが、指定されたドキュメント構造のセグメント定義に適合しているかどうかが確認されます。コンポーネントの検証では、メッセージ内のデータ構造が、ドキュメント構造のデータ構造定義に適合しているかどうかが確認されます。コード・テーブルの検証では、メッセージ内のコード・テーブルがドキュメント構造内のコード・テーブルに一致するかどうかが確認されます。

検証が完了すると、以下のようなレポートが表示されます。

```
Validating messages ..
72 messages: 18-ok/54-failed (was 34/38) - 25%/75% (was 47%/53%)
105 auto-fixable validation error(s) found
```

1.1.4.1 カスタム・スキーマの修正

検証後、メッセージ・アナライザによって特定された実行可能な修正を行うかどうかを尋ねるプロンプトが表示されます。実行可能な修正が複数ある場合は、個々にドリルダウンして、実行する修正を選択するよう求めるプロンプトが表示されます。修正を選択すると、その修正はすぐには実行されずキューに追加されます。修正を選択し終わったら、キュー内のすべての修正を受け入れるかどうかを尋ねるプロンプトが表示されます。

実行可能な修正のサブセットを受け入れる場合は、プロンプトを使用して、実行する修正を指定し、修正を適用するかどうかを尋ねるプロンプトが表示されるまで ^ を入力します。

修正のカテゴリの選択

複数のタイプの検証 (フィールド、コンポーネント、およびコード・テーブル) を実行した場合、実行可能な修正が複数のカテゴリに対して検出される可能性があります。この場合、どのタイプの修正を実行するかを指定するよう求めるプロンプトが表示されます。実行可能な修正が 1 つのカテゴリ (セグメント、データ構造、またはコード・テーブル) でのみ見つかった場合、このプロンプトはスキップされます。

```
Select a category of fixes to process
 [1] 70 code table(s)
 [2] 7 data structure(s)
 [3] 28 segment structure(s)
Enter one of the above options, ^ to go back a level:
```

構造、セグメント、またはテーブルの選択

複数のセグメント、データ構造、またはコード・テーブルに対して実行可能な修正がある場合、実行する修正を選択するよう求めるプロンプトが表示されます。修正の 1 つを選択すると、その修正はリストから削除されて、別の修正を選択できます。修正の選択が完了し、一部の修正を受け入れたくない場合は、修正を受け入れるためのプロンプトが表示されるまで、^ を入力します。

例えば、複数のデータ構造に対する修正があるとします。以下のようなプロンプトが表示されます。

```
Select a data structure to process
 [1] 2.4:CE coded element (1)
 [2] 2.4:HD hierarchic designator (1)
 [3] 2.4:ID coded value for HL7 defined tables (1)
 [4] 2.4:IS coded value for user-defined tables (1)
 [5] 2.4:MSG Message Type (1)
 [6] 2.4:PL person location (1)
 [7] 2.4:ST string data (1)
Enter one of the above options, ^ to go back a level:
```

キューへの修正の追加

ほとんどの場合、セグメント、データ構造、またはコード・テーブルを選択するたびに、修正を受け入れるかどうかを尋ねるプロンプトが表示されます。Y を入力すると、修正はすぐには適用されず、修正のキューに追加され、最終確認を行ったときに適用されます。同じコード・テーブル、セグメント、またはデータ構造に対する修正が複数ある場合は、修正をキューに追加する前に、実行する修正を選択するよう求めるプロンプトが表示されます。

修正の実行

実行可能な修正をすべてキューに追加するか、^ を使用して修正の追加を完了したことを示すと、キューに入っている修正をカスタム・スキーマに適用するかどうかを尋ねるプロンプトが表示されます。まだカスタム・スキーマに存在しない項目は、変更を実行する前に、組み込みのライブラリ・スキーマからコピーされます。

例えば、以下のような変更のキューをカスタム・スキーマに適用するかどうかを尋ねるプロンプトが表示される場合があります。

```
Items queued for fixes to HL7 configuration:
 *Do you want to add code 'Z' to code table 2.4:1 (Administrative sex)
 *Do you want to add code 'EC' to code table 2.4:131 (!Copied from 2.5 - Contact Role)
 *Do you want to add 3 dummy components to data structure 2.4:CE (coded element)
 *Do you want to add 1 dummy components to data structure 2.4:IS (coded value for user-defined tables)

 *Do you want to increase the maximum size for field 2 (EncodingCharacters) in segment structure 2.4:MSH
 (Message Header) from 4 to 5
 *Do you want to make field 7 (DateTimeOfMessage) in segment structure 2.4:MSH (Message Header) optional

Fixes marked with a '*' apply to library items, which cannot be updated directly
Do you want to apply these fixes? (Y/N):
```

変更の理解

関連する修正を実行する (例えば、新しいコードをコード・テーブルに追加する) 前に、メッセージ・アナライザは以下を実行しなければならない場合があります。

- 項目 (例えば、コード・テーブル) をカスタム・スキーマにコピーする
- カスタム項目を、以前に参照していたライブラリ項目ではなく他のカスタム項目を参照するように更新する

これらの変更は、修正を適用することを確認した後、更新レポートに詳細に記述されます。コピー、参照の更新、またはプリンシパルの変更はそれぞれ専用の行に詳細に記述されます。

`Path:` で始まる行には、ドキュメント構造、セグメント構造、データ構造、およびコード・テーブルの更新の概要が記述されます。このような行の + は、その項目がカスタム・スキーマにコピーされたことを意味し、* は、指定された参照が更新されたことを意味します。

例えば、`ORU_R01 --> PID -*-> XTN+ -*-> 202+` のパスは以下を意味します。

- カスタム・ドキュメント構造 ORU_R01

- カスタム・セグメント PID : XTN への参照が更新された
- データ構造 XTN : カスタム・スキーマにコピーされ、202 への参照が更新された
- コード・テーブル 202 : カスタム・スキーマにコピーされた

パスには、検証で特定されたプリンシパル更新ターゲットは記述されません。上の例では、これは、コード・テーブルへのコードの追加になります。

1.1.5 修正の履歴

メイン・メニューから [( )] オプションを選択すると、メッセージ・アナライザが HL7 構成のカスタム・スキーマに対して実行した変更のリストを生成できます。オプションの 1 つを選択して、プロンプトに従います。履歴をファイルに書き込むためのオプションが表示されます。

```
Workflow/2 History of Configuration Changes
 [1] List all changes
 [2] List changes made today
 [3] List changes for a range of dates
Enter one of the above options, ^ to go back a level:
```

1.1.6 ワークスペースの設定

メイン・メニューから [ ] オプションを選択すると、新しいワークスペースの作成、新しい一連のメッセージのインポート、またはスキーマやドキュメント構造の変更を行うことができます。メッセージ・アナライザが再び起動し、初めて起動する場合と同じように、ワークスペース、メッセージ・ソース、スキーマ、およびドキュメント構造を指定するよう求めるプロンプトが表示されます。既存のワークスペースのパスを入力した場合、現在の設定で続行するか、それともパスをクリアして新しい一連のメッセージをそこにコピーするかを尋ねられます。

2
プロダクション・ジェネレータ

プロダクション・ジェネレータを使用すると、CSV ファイルの HL7 インタフェース・ルートをすべて入力して、各ノードに必要なすべてのコンポーネント（サービス、ルータ、ルール、変換、およびオペレーション）が含まれるスケルトンの相互運用プロダクションを生成できます。プロダクションが生成された後に、まだニーズに合うようにプロダクションを微調整する必要はありますが、プロダクション・ジェネレータを使用すると初期の実装が迅速化されます。また、プロダクション・ジェネレータを使用して既存のプロダクションを更新することもできます。

プロダクション・ジェネレータは、CSV ファイルを使用してプロダクション・コンポーネントを作成する InterSystems ターミナルのユーティリティです。コンマまたはセミコロンで区切られたファイルを使用できますが、両方のファイルで同じ区切り文字を使用する必要があります。必要な CSV ファイルは以下のとおりです。

- 構成ファイル – 生成されるコンポーネントの名前付け規約を定義します。
- 項目ファイル – プロダクション・ジェネレータにプロダクションのコンポーネントに関する情報を提供します。

プロダクション・ジェネレータによってプロダクションに自動的に追加されるビジネス・ホスト `Ens.Activity.Operation.Local` は、アクティビティ・モニタリングのために使用され、この機能が必要ない場合は削除できます。

2.1 プロダクション・ジェネレータの実行

プロダクション・ジェネレータを実行するには、以下の手順に従います。

1. InterSystems ターミナルを開きます。
2. `set $namespace="`*`namespace`*`"` と入力して、プロダクションを作成するネームスペースに移動します。
3. 以下のように入力します。

```
set status =
##class(EnsLib.InteropTools.HL7.ProductionGenerator).Load("configuration.csv","items.csv",.out)
```

   引数は以下のとおりです。

   - `configuration.csv` は、構成ファイルのパスとファイル名です。
   - `items.csv` は、項目ファイルのパスとファイル名です。
   - `out` は、プロダクション・ジェネレータによってレポートされるデータの問題を収集します。

4. エラーをチェックするには、以下のように入力します。

```
write status
```

結果は 1 になります。

簡単なプロダクションの生成に使用できるサンプル CSV ファイルは、"CSV ファイルの例" を参照してください。

2.2 構成ファイル

構成ファイルは、生成されるプロダクションに属するコンポーネント (ビジネス・サービスや変換など) の名前付け規約を定義します。コンポーネント名は、リテラル値とプレースホルダの組み合わせから構築されます。プレースホルダの構文は { } です。プレースホルダには、プロダクション・ジェネレータによって以下の 3 つの場所からの値が入力されます。

- 構成ファイル内の他のレコード。
- 項目 CSV ファイルからの値。項目 CSV ファイルからの値をプレースホルダに入力するには、値が含まれる列名を指定します。例えば、項目ファイルに `BusinessServiceName` という名前の列があり、その列の最初のレコードに `SourceA` が指定されているとします。構成ファイルの `ServiceName` キーの値が `{BusinessServiceName}` である場合、`SourceA` という名前のサービスがプロダクション・ジェネレータによって作成されます。項目ファイルの `BusinessServiceName` 列に、値 `SourceB` が入った 2 つ目のレコードがある場合、`SourceB` という名前の別のサービスが作成されます。
- プロダクション・ジェネレータ・ユーティリティを実行するネームスペース。プロダクション・ジェネレータを `MyArea` ネームスペースで実行する場合、`{Namespace}` プレースホルダを使用するすべてのコンポーネントの名前に `MyArea` が含まれます。

以下に、必要なレコードがすべて含まれる構成ファイルの例を示します。`Key`、`Value`、および `Description` が含まれる最初のレコードは必須です。この例では、`{BusinessServiceName}` は項目ファイルの列から取得され、`{Namespace}` は、プロダクション・ジェネレータ・ユーティリティを実行したネームスペースから取得されます。

キー	値	説明
ProductionClass-Name	MyBasePackage.MyProduction	生成されるプロダクションの名前。ここでは `MyBasePackage.MyProduction` になります。
ServiceName	{BusinessServiceName}	サービスの名前付け規約を定義します。
RouterName	{BusinessServiceName}Router	ルータ・プロセスの名前付け規約を定義します。
OperationName	{BusinessOperationName}	オペレーションの名前付け規約を定義します。
RuleName	{BasePackage}.{Namespace}.Rules.{Business-ServiceName}Rule	ルールの名前付け規約を定義します。
Transformation-Name	{BasePackage}.{Namespace}.Transforms.{Business-ServiceName}Transform	変換の名前付け規約を定義します。

これらの必須のレコードのほかに、以下のオプションのレコードを構成ファイルに追加することができます。

キー	説明
ConstrainRule	既定値は 1 です。つまり、ルールの定義で、constraint フィールドによって "source" がビジネス・サービスの名前に設定されることを意味します。他の制約は設定されません。0 に設定すると、ルールの constraint フィールドは空のままです。
Override_componentID:setting	componentID:setting に、項目ファイルの指定した列からの値を入力します。例えば、Key が `Override_S:H:MessageSchemaCategory`、Value が {SourceDocVersion} の場合、[メッセージ・スキーマ・カテゴリ] 設定は、項目ファイルの SourceDocVersion 列の値に設定されます。これにより、この設定の値を常に既存の列と共有する場合は、構成ファイルに列を追加しなくても済みます。 または、Value 列に固定値を指定できます。これは、項目ファイルに componentID:setting という名前の列を追加して、その列のすべてのレコードに同じ固定値を入力することと同じです。 構文 componentID:setting の詳細は、"コンポーネント設定の指定" を参照してください。

構成ファイルの他の値に名前を付ける目的のためだけに、構成ファイルに行を定義することもできます。これらの場合、キーには任意のユーザ定義文字列を指定できます。例えば、構成ファイルに以下が含まれているとします。

キー	値	説明
EnvironmentType	Test	ProductionClassName の値を入力するために使用されるオプションのレコード。
ProductionClassName	MyProduction.{EnvironmentType}	生成されるプロダクションの名前。ここでは `MyProduction.Test` になります。

2.3 項目ファイル

項目 CSV ファイルの各レコードは、生成されるプロダクションのインタフェース・ルートに対応しており、どのコンポーネントをその属性で作成するかを決定します。複数のルートで同じコンポーネントを使用する場合は、1 つのコンポーネントが作成されます。例えば、同じビジネス・サービスが複数のルートに属する場合、プロダクションでは 1 つのビジネス・サービスのみが作成されます。

項目 CSV ファイル内の列の順序は、プロダクション・ジェネレータの動作には影響しません。各列は、以下のいずれかを実行します。

- コンポーネントの設定を指定する。
- コンポーネントのタイプを指定する。
- プロダクション・ジェネレータに特別な命令を出す。
- 構成 CSV ファイルのプレースホルダの値を提供する。これらの列名に特殊文字や句読点文字を使用することはできません。

2.3.1 コンポーネント設定の指定

項目ファイルを使用して、コンポーネントの設定の初期値を指定できます。ほとんどの設定は、列名に *componentID*:*settingName* という特殊な構文を使用して指定します。例えば、オペレーションの [**有効**] 設定は、列名 `O:Enabled` で指定します。componentID は、以下のいずれかにできます。

ComponentID	コンポーネント
S:	ビジネス・サービス
S:A:	サービス・アダプタ
S:H:	サービス・ホスト
O:	ビジネス・オペレーション
O:A:	オペレーション・アダプタ
O:H:	オペレーション・ホスト
P:	ルータ・ビジネス・プロセス
R:	ルール
T:	変換

settingName は、定義するコンポーネント設定の名前です。正しい名前は、以下の 2 つの方法で取得できます。

- この設定が定義されたプロダクションが既にある場合は、そのプロダクション・クラスをスタジオまたは Atelier で開いて、設定名を検索します。“ホスト” または “アダプタ” のどちらを指定する必要があるのか、あるいはどちらも指定する必要がないのかを確認できます。
- また、管理ポータルでダミー・プロダクションを作成してコンポーネントを追加することもできます。その後、右側のパネルの設定に移動し、システムの既定の設定のアイコンをクリックして、名前とホスト/アダプタを表示します。英語版以外のシステムを使用している場合は、マウスでローカル名をポイントすると、英語の設定名を表示できます。リストされるのはホストとアダプタの値だけです。

複数の値を指定できる設定の場合は、レコード内の値をコンマで区切ります。その際、スペースは使用しません。例えば、新しいサービスの [**カテゴリ**] 設定を `Test1` および `Test2` に設定するには、項目ファイルの `S:Category` 列に `Test1,Test2` と指定します。区切り文字としてコンマを使用して、テキスト・エディタで CSV ファイルを手動で編集する場合は、`"Test1,Test2"` と指定します。

設定は以下のいずれかで定義されます。

- コンポーネントのクラス定義
- コンポーネントのアダプタ・クラス定義
- これらのクラス定義のスーパークラス
- クラス Ens.Config.Item (例えば、[有効] 設定または [プール・サイズ] 設定)

2.3.1.1 HL7 メッセージ設定

ソースおよびターゲットの HL7 メッセージの特性を定義する設定では、*componentID*:*settingName* の構文は使用されません。代わりに、`SourceType`、`SourceSchema`、`TargetType`、および `TargetSchema` の各列を使用して、ソースおよびターゲットの HL7 メッセージのタイプ (例えば、ADT_A01) とスキーマ (例えば、2.7.1) を指定することができます。

2.3.2 コンポーネントのタイプの指定

項目ファイルには、プロダクション・ジェネレータによって作成されるビジネス・サービスとビジネス・オペレーションのタイプを指定する、列 ServiceType および OperationType が含まれる必要があります。ServiceType および OperationType で指定できる値は、TCP、File、FTP、HTTP、および SOAP です。サービスまたはオペレーションの基盤となるクラスは、ServiceType または OperationType で指定された値に基づきます。ServiceType 列または OperationType 列で MyProduct.MyFileService のような完全修飾クラス名を指定することにより、サービスまたはオペレーションに対してカスタム・クラスを指定できます。

オプションで、列 RouterType を項目ファイルに含め、これによってインタフェース・ルートに関連付けられたカスタム・ルータのクラス名を指定できます。この列がない場合は、クラス EnsLib.HL7.MsgRouter.RoutingEngine が使用されます。

2.3.3 特別な命令

事前定義済みの特定の列名は、プロダクション・ジェネレータに特殊な命令を提供します。この列のレコードのブーリアン値 (0 または 1) は、プロダクション・ジェネレータがその命令を実行する必要があるかどうかを示します。以下の列を使用して特殊な命令を指定できます。

列名	説明
RuleDisabled	ルールを有効にする場合は値を 0 に設定し、ルールを無効にする場合は 1 に設定します。この命令がない場合、ルールは無効になります。
NoTransformation	プロダクション・ジェネレータが変換を生成しないようにする場合は、値を 1 に設定します。プロダクションを更新する場合、この列を追加しても既存の変換は削除されません。
T:Create	既定では、変換の送信メッセージは受信メッセージのコピーです。T:Create 列を追加してその値を new に設定すると、変換の送信メッセージは空のオブジェクトになります。

2.4 CSV ファイルの例

以下に、連携して基本的なプロダクションを作成する構成 CSV ファイルと項目 CSV ファイルの例を示します。以下をコピーして CSV テキスト・ファイルに貼り付け、プロダクション・ジェネレータに入力することができます。また、ファイルをスプレッドシート・アプリケーションで開いて列を表示することもできます。プロダクション・ジェネレータでは、セミコロンも区切り文字として使用できます。

構成ファイル

```
Key,Value,Description
BasePackage,MyPackage,Base package name of all classes
ProductionClassName,{BasePackage}.MyProduction,Definition of the production name
ServiceName,{ServiceName},Definition of service name
RouterName,{ServiceName}_Router,Definition of process name
OperationName,{OperationName},Definition of operation name
RuleName,{BasePackage}.{Namespace}.Rules.{ServiceName}Rule,Definition of rule name
TransformationName,{BasePackage}.{Namespace}.Transforms.{ServiceName}Transform,Definition of
transformation name
```

項目ファイル

```
ServiceName,OperationName,ServiceType,OperationType,O:Enabled,S:A:FileSpec,S:H:MessageSchemaCategory,O:H:FailureTimeout,P:Category,NoTransformation
Service1,Op1,File,File,0,*,2.7.1,60,Cat1,0
Service2,Op2,TCP,FTP,1,,2.7.1,,,1
```

2.5 既存のプロダクションの更新

CSV ファイルを変更した後でプロダクション・ジェネレータを再実行すると、管理ポータルを使用せずに、生成されたコンポーネントを修正できます。設定の値をクリアするには、設定の列のレコードで `""` を使用します。以前に値に区切り文字が含まれていた列（例えば、`Categories` 列の `FTP,Hospital`）があると、値をクリアしようとしたときに問題が発生します。この場合は、`"""""""` を使用して値をクリアする必要があります。NULL データが含まれるレコードでは、生成されるコンポーネントは変更されません。

重要なロジックが失われるのを防ぐため、特定の項目は、プロダクションを更新する際にプロダクション・ジェネレータによって削除されません。

- 項目ファイルから行を削除する場合、その行で定義されているサービスとオペレーションはプロダクションから削除されません。
- 変換が存在していて、ルールに割り当てられている場合、NoTransformation 列を項目ファイルに追加すると、変換は削除されません。
- いったん作成されたルーティング・ルールは削除されません。

3

HL7 移行ツール

HL7 移行ツールでは、その他の製品で開発されたルックアップ・テーブルと変換をインターシステムズのルックアップ・テーブルと DTL クラスに変換できます。元の変換が HL7 メッセージを取り込み、それを別の HL7 メッセージに変換するように設計されている場合にのみ、HL7 移行ツールが機能します。

現在、以下からの変換を移行できます。

- Cloverleaf
- eGate
- DataGate

どの場合でも、JRE をインストール済みであることを最初に確認します。

3.1 前提条件 : Java 環境の設定

HL7 移行ツールを設定または実行する前に、以下の手順に従う必要があります。

1. Java ランタイム環境 (JRE) の最新バージョンをインストールします。システムに複数バージョンの JRE をインストールしておき、最新バージョンへのパスを使用して、このガイドの手順を実行できます。
2. 有効なバージョンが JAVA_HOME 環境変数に反映されていない場合は、必要に応じて以下の手順に従い、Java JRE のフル・パスを指定します。

3.2 Cloverleaf

移行ツールでは、Cloverleaf のテーブルと変換をインターシステムズのルックアップ・テーブルと DTL クラスに変換できます。Cloverleaf 変換が HL7 メッセージを取り込み、それを別の HL7 メッセージに変換するように設計されている場合にのみ、移行ツールは機能します。

移行ツールは、基本的な変換ロジックを DTL クラスに変更するために設計されており、Cloverleaf のカスタム・コード、ループ、およびルーティング・ルールを処理するためのものではありません。基本的な TCL コマンドは変換されますが、ループなどのほとんどのコマンドは、移行後に DTL クラスに追加する必要があります。自動的には移行されないコードは、DTL コードにコメントとして挿入されます。

3.2.1 移行ツールの設定

移行ツールを実行する前に、インターシステムズ製品で作業用のネームスペースを作成して、セットアップ・メソッドを実行する必要があります。

1. InterSystems ターミナルを開きます。
2. 以下のコマンドを入力して HSLIB ネームスペースに変更し、移行用の新しいネームスペースを作成して、その新しいネームスペースに変更します。

```
Set $namespace = "HSLIB"
Do ##class(HS.Util.Installer.Foundation).Install("MyNamespace")
Set $namespace = "MyNamespace"
```

3. 設定ツールを実行します。

```
Do ##class(EnsLib.InteropTools.HL7.Setup).Run()
```

4. 設定ツールのプロンプトに回答します。

プロンプト	入力内容
Please enter a valid type	Cloverleaf
クラス名	設定ツールで定義された設定を格納するために使用する新しいクラス名。これは、移行を実行するために使用されるクラスです。例えば、`MyPkg.MyCloverleaf` です。
Java クラスパス	ANTLR Jar ファイルのパスとファイル参照。例えば、`c:\Migration\antlr-4.7.2-complete.jar` のようになります。
Java 実行可能ファイルのパス	マシン上にある Java 実行可能ファイルのパスとファイル参照。例えば、`c:\java\jdk-13.0.1\bin\java.exe` のようになります。
Java (*java) ファイルのパス	*java ファイルのパス。必ず末尾の \ を含めてください。例えば、`c:\Migration\Cloverleaf\java\` のようになります。

または、以下のような方法でパラメータを設定ツールに渡すこともできます。

```
set status =  ##class(EnsLib.InteropTools.HL7.Setup).Run("Cloverleaf","MyPkg.MyCloverleaf",
"c:\Migration\antlr-4.7.2-complete.jar",
"c:\java\jdk-13.0.1\bin\java.exe")
```

このコマンドが失敗した場合は、`Do $SYSTEM.Status.DisplayError(status)` と入力するとエラーを表示できます。

3.2.2 移行ツールの実行

Cloverleaf 変換ファイルはデータ・ルックアップ・テーブルを参照できるため、変換ファイルを変換する前に、すべての Cloverleaf ルックアップ・テーブルを変換する必要があります。

3.2.2.1 ルックアップ・テーブルの変換

Cloverleaf 環境によっては、変換前に、変換するルックアップ・テーブル (`*.tbl`) が複数存在する場合があります。各ルックアップ・テーブルは、インターシステムズ製品の永続化ルックアップ・テーブルに変換されます。1 つの *.tbl ファイ

ル、または複数のファイルが含まれる 1 つのディレクトリに対して移行ツールを実行できます。Cloverleaf ルックアップ・テーブルを変換するには、以下の手順に従います。

1. InterSystems ターミナルで、移行用に作成したネームスペースに変更します。以下に例を示します。

```
Set $namespace="MyNamespace"
```

2. 一度に複数の *.tbl ファイルを変換する場合は、ファイルを 1 つのディレクトリに配置して、以下を入力します。

```
set status = ##class(MyPkg.MyCloverleaf).TableImport("c:\migration\tables\","CL.Lookup.")
```

引数は以下のとおりです。

- `MyPkg.MyCloverleaf` は、移行ツールの設定時に指定したクラスの名前です。
- `c:\migration\tables\` には、Cloverleaf ファイルが含まれます。必ず末尾の `\` を付けてください。
- `CL.Lookup.` は、ルックアップ・コードで作成されたインターシステムズのテーブルの接頭語です。例えば、Cloverleaf ファイルが `codes.tbl` である場合、インターシステムズのルックアップ・テーブルは `CL.Lookup.codes` になります。このパラメータには任意の有効なパッケージ名を指定できますが、必ず末尾のピリオド (`.`) を付けてください。

変換が成功したかどうかを確認するには、`write status` と入力します。成功した場合は、`1` が表示されます。

1 つの Cloverleaf ルックアップ・ファイルに対して移行ツールを実行するには、以下のように入力します。

```
set status = ##class(MyPkg.MyCloverleaf).TableImport("c:\migration\tables\input.tbl","CL.Lookup.")
```

3.2.2.2 変換の変換

HL7 変換で使用されるルックアップ・テーブルを変換したら、Cloverleaf 変換ファイルを DTL クラスに変換できます。各 Cloverleaf 変換ファイル (*.xlt) に対して、以下のように入力します。

1. InterSystems ターミナルで、移行用に作成したネームスペースに変更します。以下に例を示します。

```
Set $namespace="MyNamespace"
```

2. 各 Cloverleaf 変換ファイルに対して、以下のように入力します。

```
set status =
##class(MyPkg.MyCloverleaf).CodeWalk("c:\migration\transforms\cloverleaf_sample.xlt","/ClassName=Sample.DTL
 /SourceDocType=2.7:ACK /TargetDocType=MyCustom2.7:ACK /TableGroupName=CL.Lookup.
/Reprocess=1",.tOutput)
```

引数は以下のとおりです。

- `MyPkg.MyCloverleaf` は、移行ツールの設定時に指定したクラスの名前です。
- `c:\migration\transforms\cloverleaf_sample.xlt` は、Cloverleaf 変換ファイルです。
- `/ClassName` では、移行ツールによって作成された新しい DTL クラスの名前を指定します。
- `/SourceDocType` は、受信 HL7 メッセージのスキーマおよび DocType 構造です。利用可能なスキーマと DocType 構造を表示するには、管理ポータルを開き、**[相互運用性]→[相互運用]→[HL7 v2.x]→[HL7 v2.x スキーマ構造]** に移動します。インターシステムズ製品にスキーマを追加する必要がある場合は、メッセージ・アナライザが役立ちます。
- `/TargetDocType` は、ターゲットのスキーマおよび DocType 構造です。`SourceDocType` は、`TargetDocType` に変換されます。

- `/TableGroupName` は、Cloverleaf ルックアップ・テーブル・ファイルの移行時に指定したルックアップ・テーブルの接頭語の名前です。変換でルックアップ・テーブルを使用しない場合は、このパラメータを省略できます。
- `/Reprocess` では、既存の DTL クラスを上書きするかどうかを指定します。`/Reprocess=1` の場合、クラスは上書きされます。
- `.tOutput` は、移行ツールで生成される情報を収集します。

`write status` と入力すると、エラーがないか確認できます。変換が成功した場合は、ターミナルに `1` が表示されます。

既定では、新しい DTL は、移行ツールによって作成された後にコンパイルされます。DTL がコンパイルされないようにするには、`CodeWalk` メソッドのパラメータとして `/BuildDTL=0` を指定します。

3.3 eGate

移行ツールでは、eGate のテーブルと変換をインターシステムズのルックアップ・テーブルと DTL クラスに変換できます。eGate 変換が HL7 メッセージを取り込み、それを別の HL7 メッセージに変換するように設計されている場合にのみ、移行ツールは機能します。

移行ツールは、基本的な変換ロジックを DTL クラスに変更するために設計されており、カスタム・コード、ループ、およびルーティング・ルールを処理するためのものではありません。自動的には移行されないコードは、DTL コードにコメントとして挿入されます。

3.3.1 移行ツールの設定

移行ツールを実行する前に、インターシステムズ製品で作業用のネームスペースを作成して、セットアップ・メソッドを実行する必要があります。

1. InterSystems ターミナルを開きます。
2. 以下のコマンドを入力して `HSLIB` ネームスペースに変更し、移行用の新しいネームスペースを作成して、その新しいネームスペースに変更します。

```
Set $namespace = "HSLIB"
Do ##class(HS.Util.Installer.Foundation).Install("MyNamespace")
Set $namespace = "MyNamespace"
```

3. 設定ツールを実行します。

```
Do ##class(EnsLib.InteropTools.HL7.Setup).Run()
```

4. 設定ツールのプロンプトに回答します。

プロンプト	入力内容
Please enter a valid type	eGate
クラス名	設定ツールで定義された設定を格納するために使用する新しいクラス名。これは、移行を実行するために使用されるクラスです。例えば、MyPkg.MyEGate のようになります。
Java クラスパス	ANTLR Jar ファイルのパスとファイル参照。例えば、c:\Migration\antlr-4.7.2-complete.jar のようになります。
Java 実行可能ファイルのパス	マシン上にある Java 実行可能ファイルのパスとファイル参照。例えば、c:\java\jdk-13.0.1\bin\java.exe のようになります。
Java (*.java) ファイルのパス	*.java ファイルのパス。必ず末尾の \ を含めてください。例えば、c:\Migration\eGate\java\ のようになります。

または、以下のような方法でパラメータを設定ツールに渡すこともできます。

```
set status =  ##class(EnsLib.InteropTools.HL7.Setup).Run("eGate","MyPkg.MyEGate",
"c:\Migration\antlr-4.7.2-complete.jar",
"c:\java\jdk-13.0.1\bin\java.exe")
```

このコマンドが失敗した場合は、Do $SYSTEM.Status.DisplayError(status) と入力するとエラーを表示できます。

3.3.2 移行ツールの実行

eGate 変換ファイルはデータ・ルックアップ・テーブルを参照できるため、変換ファイルを変換する前に、すべての eGate ルックアップ・テーブルを変換する必要があります。

3.3.2.1 ルックアップ・テーブルの変換

eGate 環境によっては、変換前に、変換するルックアップ・テーブルが複数存在する場合があります。各ルックアップ・テーブルは、インターシステムズ製品の永続化ルックアップ・テーブルに変換されます。1 つのファイル、または複数のファイルが含まれる 1 つのディレクトリに対して移行ツールを実行できます。eGate ルックアップ・テーブルを変換するには、以下の手順に従います。

1. InterSystems ターミナルで、移行用に作成したネームスペースに変更します。以下に例を示します。

   ```
   Set $namespace="MyNamespace"
   ```

2. 一度に複数のファイルを変換する場合は、ファイルを 1 つのディレクトリに配置して、以下を入力します。

   ```
   set status = ##class(MyPkg.MyEGate).TableImport("c:\migration\tables\","eGate.Lookup.","*.txt")
   ```

   以下は値の説明です。

   - MyPkg.MyEGate は、移行ツールの設定時に指定したクラスの名前です。
   - c:\migration\tables\ には、eGate ルックアップ・テーブル・ファイルが含まれます。必ず末尾の \ を付けてください。

- `eGate.Lookup.` は、ルックアップ・コードで作成されたインターシステムズのテーブルの接頭語です。例えば、eGate ファイルが `codes.txt` である場合、インターシステムズのルックアップ・テーブルは `eGate.Lookup.codes` になります。このパラメータには任意の有効なパッケージ名を指定できますが、必ず末尾のピリオド (`.`) を付けてください。
- `*.txt` は、eGate ルックアップ・テーブル・ファイルのファイル拡張子を指定します。

変換が成功したかどうかを確認するには、`write status` と入力します。成功した場合は、`1` が表示されます。

1 つの eGate ルックアップ・ファイルに対して移行ツールを実行するには、以下のように入力します。

```
set status = ##class(MyPkg.MyEGate).TableImport("c:\migration\tables\input.txt","eGate.Lookup.")
```

3.3.2.2 変換の変換

HL7 変換で使用されるルックアップ・テーブルを変換したら、eGate 変換ファイルを DTL クラスに変換できます。各 eGate 変換ファイル (*.tsc) に対して、以下のように入力します。

1. InterSystems ターミナルで、移行用に作成したネームスペースに変更します。以下に例を示します。

   ```
   Set $namespace="MyNamespace"
   ```

2. 各 eGate 変換ファイルに対して、以下のように入力します。

   ```
   set status =
   ##class(MyPkg.MyEGate).CodeWalk("c:\migration\transforms\eGate_sample.tsc","/ClassName=Sample.DTL
    /SourceDocType=2.7:ACK /TargetDocType=MyCustom2.7:ACK /TableGroupName=eGate.Lookup.
   /Reprocess=1",.tOutput)
   ```

   以下は値の説明です。

   - `MyPkg.MyEGate` は、移行ツールの設定時に指定したクラスの名前です。
   - `c:\migration\transforms\eGate_sample.tsc` は、eGate 変換ファイルです。
   - `/ClassName` では、移行ツールによって作成された新しい DTL クラスの名前を指定します。
   - `/SourceDocType` は、受信 HL7 メッセージのスキーマおよび DocType 構造です。利用可能なスキーマと DocType 構造を表示するには、管理ポータルを開き、**[相互運用性]→[相互運用]→[HL7 v2.x]→[HL7 v2.x スキーマ構造]** に移動します。インターシステムズ製品にスキーマを追加する必要がある場合は、メッセージ・アナライザが役立ちます。
   - `/TargetDocType` は、ターゲットのスキーマおよび DocType 構造です。`SourceDocType` は、`TargetDocType` に変換されます。
   - `/TableGroupName` は eGate ルックアップ・テーブル・ファイルの移行時に指定したルックアップ・テーブルの接頭語の名前です。変換でルックアップ・テーブルを使用しない場合は、このパラメータを省略できます。
   - `/Reprocess` では、既存の DTL クラスを上書きするかどうかを指定します。`/Reprocess=1` の場合、クラスは上書きされます。
   - `.tOutput` は、移行ツールで生成される情報を収集します。

   `write status` と入力すると、エラーがないか確認できます。変換が成功した場合は、ターミナルに `1` が表示されます。

既定では、新しい DTL は、移行ツールによって作成された後にコンパイルされます。DTL がコンパイルされないようにするには、`CodeWalk` メソッドのパラメータとして `/BuildDTL=0` を指定します。

3.4 DataGate

移行ツールでは、DataGate のテーブルと変換をインターシステムズのルックアップ・テーブルと DTL クラスに変換できます。DataGate 変換が HL7 メッセージを取り込み、それを別の HL7 メッセージに変換するように設計されている場合にのみ、移行ツールは機能します。

移行ツールは、基本的な変換ロジックを DTL クラスに変更するために設計されており、カスタム・コード、ループ、およびルーティング・ルールを処理するためのものではありません。自動的には移行されないコードは、DTL コードにコメントとして挿入されます。

3.4.1 移行ツールの設定

移行ツールを実行する前に、インターシステムズ製品で作業用のネームスペースを作成して、セットアップ・メソッドを実行する必要があります。

1. InterSystems ターミナルを開きます。
2. 以下のコマンドを入力して HSLIB ネームスペースに変更し、移行用の新しいネームスペースを作成して、その新しいネームスペースに変更します。

```
Set $namespace = "HSLIB"
Do ##class(HS.Util.Installer.Foundation).Install("MyNamespace")
Set $namespace = "MyNamespace"
```

3. 設定ツールを実行します。

```
Do ##class(EnsLib.InteropTools.HL7.Setup).Run()
```

4. 設定ツールのプロンプトに回答します。

プロンプト	入力内容
Please enter a valid type	DataGate
クラス名	設定ツールで定義された設定を格納するために使用する新しいクラス名。これは、移行を実行するために使用されるクラスです。例えば、MyPkg.MyDataGate のようになります。
Java クラスパス	ANTLR Jar ファイルのパスとファイル参照。例えば、c:\Migration\antlr-4.7.2-complete.jar のようになります。
Java 実行可能ファイルのパス	マシン上にある Java 実行可能ファイルのパスとファイル参照。例えば、c:\java\jdk-13.0.1\bin\java.exe のようになります。
Java (*java) ファイルのパス	*java ファイルのパス。必ず末尾の \ を含めてください。例えば、c:\Migration\DataGate\java\ のようになります。

または、以下のような方法でパラメータを設定ツールに渡すこともできます。

```
set status =  ##class(EnsLib.InteropTools.HL7.Setup).Run("DataGate","MyPkg.MyDataGate",
"c:\Migration\antlr-4.7.2-complete.jar",
"c:\java\jdk-13.0.1\bin\java.exe")
```

このコマンドが失敗した場合は、`Do $SYSTEM.Status.DisplayError(status)` と入力するとエラーを表示できます。

3.4.2 移行ツールの実行

DataGate 変換ファイルはデータ・ルックアップ・テーブルを参照できるため、変換ファイルを変換する前に、すべての DataGate ルックアップ・テーブルを変換する必要があります。

3.4.2.1 ルックアップ・テーブルの変換

DataGate 環境によっては、変換前に、変換するルックアップ・テーブルが複数存在する場合があります。各ルックアップ・テーブルは、インターシステムズ製品の永続化ルックアップ・テーブルに変換されます。1 つのファイル、または複数のファイルが含まれる 1 つのディレクトリに対して移行ツールを実行できます。DataGate ルックアップ・テーブルを変換するには、以下の手順に従います。

1. InterSystems ターミナルで、移行用に作成したネームスペースに変更します。以下に例を示します。

   ```
   Set $namespace="MyNamespace"
   ```

2. 一度に複数のファイルを変換する場合は、ファイルを 1 つのディレクトリに配置して、以下を入力します。

   ```
   set status = ##class(MyPkg.MyDataGate).TableImport("c:\migration\tables\","DataGate.Lookup.","*.txt")
   ```

   以下は値の説明です。

   - `MyPkg.MyDataGate` は、移行ツールの設定時に指定したクラスの名前です。
   - `c:\migration\tables\` には、DataGate ルックアップ・テーブル・ファイルが含まれます。必ず末尾の `\` を付けてください。
   - `DataGate.Lookup.` は、ルックアップ・コードで作成されたインターシステムズのテーブルの接頭語です。例えば、DataGate ファイルが `codes.txt` である場合、インターシステムズのルックアップ・テーブルは `DataGate.Lookup.codes` になります。このパラメータには任意の有効なパッケージ名を指定できますが、必ず末尾のピリオド (`.`) を付けてください。
   - `*.txt` は、DataGate ルックアップ・テーブル・ファイルのファイル拡張子を指定します。

   変換が成功したかどうかを確認するには、`write status` と入力します。成功した場合は、`1` が表示されます。

   1 つの DataGate ルックアップ・ファイルに対して移行ツールを実行するには、以下のように入力します。

   ```
   set status = ##class(MyPkg.MyDataGate).TableImport("c:\migration\tables\input.txt","DataGate.Lookup.")
   ```

3.4.2.2 変換の変換

HL7 変換で使用されるルックアップ・テーブルを変換したら、DataGate 変換ファイルを DTL クラスに変換できます。各 DataGate 変換ファイル (*.tsc) に対して、以下のように入力します。

1. InterSystems ターミナルで、移行用に作成したネームスペースに変更します。以下に例を示します。

   ```
   Set $namespace="MyNamespace"
   ```

2. 各 DataGate 変換ファイルに対して、以下のように入力します。

```
set status =
##class(MyPkg.MyDataGate).CodeWalk("c:\migration\transforms\DataGate_sample.tsc","/ClassName=Sample.DTL
 /SourceDocType=2.7:ACK /TargetDocType=MyCustom2.7:ACK /TableGroupName=DataGate.Lookup.
/Reprocess=1",.tOutput)
```

以下は値の説明です。

- `MyPkg.MyDataGate` は、移行ツールの設定時に指定したクラスの名前です。
- `c:\migration\transforms\DataGate_sample.tsc` は、DataGate 変換ファイルです。
- `/ClassName` では、移行ツールによって作成された新しい DTL クラスの名前を指定します。
- `/SourceDocType` は、受信 HL7 メッセージのスキーマおよび DocType 構造です。利用可能なスキーマと DocType 構造を表示するには、管理ポータルを開き、**[相互運用性]→[相互運用]→[HL7 v2.x]→[HL7 v2.x スキーマ構造]** に移動します。インターシステムズ製品にスキーマを追加する必要がある場合は、メッセージ・アナライザが役立ちます。
- `/TargetDocType` は、ターゲットのスキーマおよび DocType 構造です。`SourceDocType` は、`TargetDocType` に変換されます。
- `/TableGroupName` は DataGate ルックアップ・テーブル・ファイルの移行時に指定したルックアップ・テーブルの接頭語の名前です。変換でルックアップ・テーブルを使用しない場合は、このパラメータを省略できます。
- `/Reprocess` では、既存の DTL クラスを上書きするかどうかを指定します。`/Reprocess=1` の場合、クラスは上書きされます。
- `.tOutput` は、移行ツールで生成される情報を収集します。

`write status` と入力すると、エラーがないか確認できます。変換が成功した場合は、ターミナルに `1` が表示されます。

既定では、新しい DTL は、移行ツールによって作成された後にコンパイルされます。DTL がコンパイルされないようにするには、`CodeWalk` メソッドのパラメータとして `/BuildDTL=0` を指定します。