# HL7 Productivity Tools

Version 2024.1
2024-05-02

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

*HL7 Productivity Tools*
InterSystems Version 2024.1 2024-05-02

For Support questions about any InterSystems products, contact:

**InterSystems Worldwide Response Center (WRC)**

| | |
|---|---|
| Tel: | +1-617-621-0700 |
| Tel: | +44 (0) 844 854 2917 |
| Email: | support@InterSystems.com |

# Table of Contents

# 1

# HL7 Message Analyzer

The Message Analyzer is a command-line utility that allows you to:

- Scan HL7 messages and compare them to a given document structure, deriving a new document structure that matches the messages. Most commonly, the derived document structure contains Z segments found in the scanned messages.
- Validate messages against a given document structure, optionally updating the custom schema's segments, data structures, and code tables to allow failing messages.

The Message Analyzer uses the following terminology:

| Term | Description |
|---|---|
| Document structure | A HL7 message structure such as ADT_01. In the Management Portal, a schema's document structures are listed on the **DocType Structure** tab. |
| HL7 configuration | Collectively refers to the schemas, document structures, segments, data structures, and code tables that are defined in the Management Portal. |
| Library schema | A base schema, for example 2.7.1, that is available in the Management Portal. A custom schema is based on one of these library schemas. |

## 1.1 Running the Message Analyzer

To run the Message Analyzer:

1. Open the InterSystems Terminal.
2. Change to the namespace that contains or will contain the HL7 interoperability productions that route your HL7 messages. This is the namespace where the custom schema will be stored. For example, enter:

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

3. Enter:

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

The Message Analyzer presents you with a series of prompts that lead you through the possible options. Entering ^ at any prompt goes back to the previous input or exits from the program, depending on the context. If only one option in a menu applies, the menu is not be shown and the option is automatically selected for you. When a prompt uses a `<`*`value`*`>` syntax, pressing **Enter** uses the *value* as the answer to the prompt.

## 1.1.1 Initial Setup

The first time you run the Message Analyzer, it prompts you for a workspace folder and information about the HL7 schema and document structure.

The next time you run the Message Analyzer, it skips this initial setup and starts with the main menu. If you need to switch the workspace folder, load different messages or use a different schema/document structure, choose the `Setup Workspace` option from the main menu.

### 1.1.1.1 Workspace Folder

When you run the Message Analyzer for the first time, it prompts you for a workspace folder:

```
Workspace folder:
```

Enter the name of a file-system folder (directory) that does not currently exist. This folder is used as a scratch workspace. During its initial setup, the Message Analyzer rejects an existing folder to avoid overwriting files. If you rerun the setup later, you can use the existing workspace or create a new one.

### 1.1.1.2 Source Folder

The following prompt asks you to specify the source file or folder:

```
Source file or folder:
```

Enter either a HL7 message file or a folder containing HL7 message files, which is a text file with each line representing a segment. A message file can contain any number of messages; the `MSH` segment is used to detect where a new message begins. The file can also contain a batch of messages.

### 1.1.1.3 Schema

The following prompt asks you to specify a schema:

```
Schema:
```

The entry must be one of the following:

- An existing custom schema.
- An existing library schema (for example, `2.4`). Because this schema cannot be used directly, you are prompted for the name of a new custom schema to be based on the library schema.
- A new custom schema. You are prompted for an existing library schema to base the custom version on.

When the Message Analyzer derives a new document structure, it makes changes to this custom schema based on the input messages. The Message Analyzer also updates the custom schema rather than a library schema when validating and modifying segments, data structures, and code tables to allow failing messages.

### 1.1.1.4 Document structure

The following prompt asks you to specify a document structure.

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

Enter the name of a document structure (e.g., ADT_A01) that exists in the custom schema or the name of a document structure which exists in some other schema. To choose from a list of document structures in the custom schema, enter the start of a document structure name, followed by `?`. For example, entering `A?` might list ADT_A01 and ADT_A02. If a new document structure name is entered, you are prompted to specify a library schema where it exists so it can be copied to the custom schema.

When deriving a new document structure, the Message Analyzer compares the structure of the input HL7 messages to the specified document structure. When validating messages, the Message Analyzer compares the segments, data structures, and code tables of the specified document structure to those in the messages.

In the Management Portal, document structures are listed on the **DocType Structure** tab of the schema.

## 1.1.2 Main Menu

Once the Message Analyzer has been through its initial setup at least once, the main menu appears:

```
[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
```

If you choose the Derive new document structures option, the Message Analyzer matches the input messages to the document structure specified during setup, and then derives a new document structure based on the structure of the messages.

If you choose the Validate messages, optionally fixing the configuration option, the Message Analyzer compares the input messages against the specified document structure, and then proposes changes to the schema that would allow failing messages to pass validation.

If you choose the View history of fixes to the configuration option, the Message Analyzer lists the changes that it has made to the custom schema.

If you choose the Setup Workspace option, the Message Analyzer re-runs the intitial setup process, allowing you to specify a new workspace, custom schema, or document structure.

## 1.1.3 Derive New Document Structures

The `Derive new document structure` option matches the input HL7 messages against the document structure specified during setup, and derives a new document structure based on the structure of the messages. For example, the derived document structure might include Z segments used by the messages.

Because the process of analyzing messages can be time-consuming, the `Derive new document structures` option prompts you to enter the percentage of messages to scan.

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

The results look like:

```
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
```

If a new document structure is derived, the segments of the new document structure is displayed. In this case, you are asked whether you want to update the document structure in the custom schema.

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

## 1.1.4 Validate Messages and Fix Configuration

The `Validate messages, optionally fixing the configuration` option of the Message Analyzer validates HL7 messages against the segments, data structures, and code tables of the custom schema. If these do not exist in the custom schema, then the Message Analyzer validates against the library schema on which the custom schema is based. Once validated, the Message Analyzer offers to change the custom schema based on the segments, data structures, and code tables of the input messages so the messages pass validation against the schema.

```
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:
```

Select one of the first three entries to indicate whether the Message Analyzer will perform that type of validation. If a menu line ends with "[ ]", the item will not be used in validation. If a menu line ends with "[X]", the item will be used in validation. When you have selected one or more entries, enter `4` to start validation.

Field validation checks whether the segments in the messages conform to segment definitions in the specified document structure. Component validation checks whether the data structures in the messages conform to the data structure definitions in the document structure. Code table validation checks whether the code tables in the messages match the code tables in the document structure.

When validation is complete, a report is shown:

```
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 Fixing the Custom Schema

After validation, the Message Analyzer prompts you whether you want to make the possible fixes it has identified. If there is more than one possible fix, you are prompted to drill down to individually choose which fixes you want to make. When you choose a fix, it is not executed immediately, but rather is added to a queue. When you are done selecting fixes, you are prompted whether you want to accept all the fixes in the queue.

If you want to accept a subset of the possible fixes, use the prompts to identify the fixes you want to make, then enter `^` until you are prompted to apply the fixes.

#### Selecting a Category of Fixes

If you ran more than one type of validation (field, component, and code table), it is possible that the Message Analyzer found potential fixes for more than one category. In this case, you are prompted to specify what type of fixes you want to make. If the Message Analyzer found possible fixes in only one category (segment, data structure, or code table), this prompt is skipped.

```
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:
```

#### Selecting a Structure, Segment, or Table

If there are possible fixes for multiple segments, data structures or code tables, you are prompted to select which fix you want to make. Once you select one of the fixes, it is removed from the list and you can select another fix. If you are done selecting fixes and do not want to accept them all, enter `^` until you are prompted to accept the fixes.

For example, suppose there are fixes for more than one data structure. You might see a prompt like:

```
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:
```

## Adding Fixes to the Queue

In most cases, each time you select a segment, data structure or code table, you are prompted whether you want to accept the fix. When you enter `Y`, the fix is not applied immediately; it is added to a queue of fixes that will be applied when you give a final confirmation. If there is more than one fix to the same code table, segment, or data structure, you are prompted to select which fix you want to make before adding it to the queue.

## Making the Fixes

When you have added all the possible fixes to the queue or have indicated you are done adding fixes by using `^`, you are prompted whether you want to apply the queued fixes to the custom schema. Items that are not already in the custom schema will be copied from the built-in library schema before the changes are made.

For example, you might be prompted to apply the following queue of changes to the custom schema:

```
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):
```

## Understanding the Changes

Before the relevant fix can be made (for example, adding a new code to a code table), the Message Analyzer might need to:

- Copy items (for example, code tables) to the custom schema.
- Update custom items to reference other custom items instead of the library items they formerly referenced.

These changes are detailed in the update report which follows your confirmation that the fixes should be applied. Each copy, reference-update or principal change is detailed on its own line.

Lines beginning with `Path:` summarize updates to document structures, segment structures, data structures and code tables. In such a line, a `+` means that the item was copied to the custom schema; a `*` means that the given reference was updated.

For example, a path of `ORU_R01 --> PID -*-> XTN+ -*-> 202+` means:

- custom document structure ORU_R01
- custom segment PID: reference to XTN was updated
- data structure XTN: copied to custom schema, reference to 202 was updated
- code table 202: copied to custom schema

The path does not describe the principal update target identified by the validation - in the example above this would be the adding of a code to the code table.

## 1.1.5 History of Fixes

When you select the `View history of fixes to the configuration (validation fixes only)` option from the main menu, you can generate a list of the changes the Message Analyzer has made to custom schema in your HL7 configuration. Select one of the options, then follow the prompts. You are given the option to write the history to a file.

```
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 Setup Workspace

Choosing the `Setup Workspace` option from the main menu allows you to create a new workspace, import a new set of messages or change the schema and document structure. The Message Analyzer restarts — as if for the first time — and prompts you for the workspace, message source, schema, and document structure. If you enter a path to an existing workspace, you are asked whether you want to continue with its current setup or to clear it and copy a new set of messages into it.

# 2

# HL7 Production Generator

With the Production Generator, you can enter all of your HL7 interface routes in a CSV file to produce a skeleton interoperability production that contains all of the components (services, routers, rules, transformations, and operations) needed for each route. Once generated, you still need to fine-tune the production to fit your needs, but the Production Generator speeds up the initial implementation. You can also use the Production Generator to update an existing production.

The Production Generator is an InterSystems Terminal utility that uses CSV files to create production components. These files can be delimited by commas or semi-colons, but both files need to use the same delimiter. The required CSV files are:

- Configuration file - Defines the naming conventions of the generated components.
- Items file - Provides the Production Generator with information about the production's components.

The business host `Ens.Activity.Operation.Local` that is automatically added to the production by the Production Generator is used for activity monitoring, and can be removed if you do not need this functionality.

## 2.1 Running the Production Generator

To run the Production Generator:

1. Open the InterSystems Terminal.
2. Enter `set $namespace="namespace"` to navigate to the namespace where the production will be created.
3. Enter:

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

   Where:

   - `configuration.csv` is the path and filename of your configuration file.
   - `items.csv` is the path and filename of your items file.
   - `out` captures data issues reported by the Production Generator.

4. To check for errors, enter:

   ```
   write status
   ```

   The result should be `1`.

For sample CSV files that you can use to generate a simple production, see Example CSV Files.

## 2.2 Configuration File

The configuration file defines the naming conventions of the components, such as business services or transformations, that are part of the generated production. The component names are built from a combination of literal values and placeholders; the syntax for placeholders is `{ }`. The Production Generator populates placeholders with values from three places:

- Other records in the configuration file.
- Values from the items CSV file. To populate a placeholder with values from the items CSV file, you specify the column name that contains the values. For example, suppose the items file has a column named `BusinessServiceName`, and the first record specifies `SourceA` in that column. If the value of the `ServiceName` key in the configuration file is `{BusinessServiceName}`, then the Production Generator creates a service named `SourceA`. If the items file contained a second record with the value `SourceB` under the `BusinessServiceName` column, the Production Generator would create another service called `SourceB`.
- The namespace where you run the Production Generator utility. When you run the Production Generator in the `MyArea` namespace, any component using the `{Namespace}` placeholder contains `MyArea` in its name.

The following is an example of a configuration file that contains all the required records. The first record containing `Key`, `Value`, and `Description` is required. In this example, `{BusinessServiceName}` comes from a column in the items file and `{Namespace}` comes from the namespace where you ran the Production Generator utility.

| Key | Value | Description |
|---|---|---|
| ProductionClassName | MyBasePackage.MyProduction | Name of the generated production, which will be `MyBasePackage.MyProduction`. |
| ServiceName | {BusinessServiceName} | Defines the naming convention of services. |
| RouterName | {BusinessServiceName}Router | Defines the naming convention of router processes. |
| OperationName | {BusinessOperationName} | Defines the naming convention of operations |
| RuleName | {BasePackage}.{Namespace}.Rules.{BusinessServiceName}Rule | Defines the name convention of rules. |
| TransformationName | {BasePackage}.{Namespace}.Transforms.{BusinessServiceName}Transform | Defines the naming convention of transformations. |

In addition to these required records, you can add the following optional records to the configuration file:

| Key | Description |
|---|---|
| ConstrainRule | Default is `1`, which means that in the definition of a rule, the **constraint** field sets "source" to the name of the business service. Other constraints do not get set. When set to `0`, the rule's **constraint** field is left empty. |
| Override_*componentID*:*setting* | Populates *componentID*:*setting* with the values from the specified column of the items file. For example, if the `Key` is `Override_S:H:MessageSchemaCategory` and the `Value` is `{SourceDocVersion}`, then the Message Schema Category setting will be set to the value in the `SourceDocVersion` column of the items file. This avoids having to add a column in the items file when the value of the setting is always shared with an existing column.<br>Alternatively, you can specify a fixed value in the Value column. This is the same as adding a column in the items file named *componentID*:*setting*, and then giving every record in the column the same fixed value.<br>For more information about the syntax of *componentID*:*setting*, see Specifying Component Settings. |

You can also define rows in the configuration file with the sole purpose of naming other values of the configuration file. In these cases, the Key can be any user-defined string. For example, suppose your configuration file includes:

| Key | Value | Description |
|---|---|---|
| EnvironmentType | Test | Optional record used to populate Value of ProductionClassName |
| ProductionClassName | MyProduction.{EnvironmentType} | Name of the generated production, which will be `MyProduction.Test`. |

## 2.3 Items File

Each record in the items CSV file corresponds to an interface route in the generated production, and determines which components are created along with their attributes. If multiple routes use the same component, the Production Generator creates a single component. For example, if the same business service is part of multiple routes, only one business service is created in the production.

The order of the columns in the items CSV file does not affect the Production Generator's behavior. Each column accomplishes one of the following:

- Specifies settings for a component.
- Specifies the type of a component.
- Gives the Production Generator special instructions.

- Provides values for placeholders in the configuration CSV file. These column names cannot contain special characters or punctuation.

## 2.3.1 Specifying Component Settings

The items file allows you to specify the initial value of settings for a component. Most settings are identified by using a special syntax in the column name: *`componentID`*`:`*`settingName`*. For example, the **Enabled** setting for an operation is identified with a column name `O:Enabled`. The *componentID* can be one of the following:

| *ComponentID* | Component |
|---|---|
| S: | Business Service |
| S:A: | Service Adapter |
| S:H: | Service Host |
| O: | Business Operation |
| O:A: | Operation Adapter |
| O:H: | Operation Host |
| P: | Router Business Process |
| R: | Rule |
| T: | Transformation |

The *settingName* is the name of the component setting you want to define. You can obtain the correct setting name in two ways:

- If you already have a production with this setting defined, open the production class in Studio or Atelier and find the setting name. You can see whether you have to specify “host”, “adapter” or neither.
- You can also create a dummy production in the Management Portal and add the component. Then, navigate to the setting in the right-hand pane and click on the icon for the system default settings to see the name and host/adapter. If you have a non-English system, you can point the mouse on the local name to display the English setting name. Only host and adapter values are listed.

For settings that can have multiple values, separate values in the record with commas without any spaces. For example, to set the **Category** setting of the new service to `Test1` and `Test2`, specify `Test1,Test2` under the `S:Category` column of the items file. If you are manually editing the CSV file with a text editor using comma as the delimiter, specify `"Test1,Test2"`.

Settings get defined in one of the following:

- The class definition of the component
- The adapter class definition of the component
- A superclass of these class definitions
- Class Ens.Config.Item (for example, the Enabled or PoolSize settings)

### 2.3.1.1 HL7 Message Settings

Settings that define the characteristics of source and target HL7 messages do not use the *`componentID`*`:`*`settingName`* syntax. Rather, you can specify the type (e.g., ADT_A01) and schema (e.g., 2.7.1) of the source and target HL7 messages using the `SourceType`, `SourceSchema`, `TargetType`, and `TargetSchema` columns.

## 2.3.2 Specifying the Type of Component

The items file must include columns, `ServiceType` and `OperationType`, that determine the type of business service and business operation created by the Production Generator. Possible values under `ServiceType` and `OperationType` are: TCP, File, FTP, HTTP and SOAP. The underlying class of the service or operation is based on the specified value of `ServiceType` or `OperationType`. You can specify a custom class for the service or operation by specifying a fully qualified classname such as `MyProduct.MyFileService` in the `ServiceType` or `OperationType` column.

Optionally, the items file can include a column `RouterType` that specifies the classname of a custom router associated with the interface route. Without this column, the class `EnsLib.HL7.MsgRouter.RoutingEngine` is used.

## 2.3.3 Special Instructions

Certain predefined column names provide special instructions to the Production Generator. A boolean value (0 or 1) in the record under the column indicates whether the Production Generator should execute the instruction. You can specify special instructions using the following columns:

| Column Name | Description |
|---|---|
| RuleDisabled | Set value to 0 to enable the rule or 1 to disable the rule. Without this instruction, the Production Generator disables the rule. |
| NoTransformation | Set value to 1 to prevent the Production Generator from creating a transformation. If you are updating a production, adding this column will not delete already existing transformations. |
| T:Create | By default, the outbound message of a transformation is a copy of the inbound message. If you add a `T:Create` column and set its value to `new`, the outbound message of the transformation will be an empty object. |

# 2.4 Example CSV Files

The following are examples of a configuration CSV file and items CSV file that work together to create a basic production. You can copy and paste the following into CSV text files and input them into the Production Generator. You can also open them up in a spreadsheet application to view the columns. The Production Generator also accepts files with a semi-colon as the delimiter.

### Configuration file

```
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
```

### Items file

```
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 Updating an Existing Production

You can re-run the Production Generator after making changes to the CSV files to alter the generated components without using the Management Portal. To clear out the value of a setting, use `""` in the record under the setting's column. Columns that previously included the delimiter in the value (e.g., `FTP,Hospital` under the `Categories` column) cause an issue when trying to clear out the value. In this case, you must use `""""""` to clear the value. Records with null data do not alter the generated components.

To prevent the loss of substantial logic, the Production Generator will not delete certain items when you update a production:

- If you remove a row from the items file, the services and operations defined in that row will not be deleted from the production.
- Once a transformation exists and is assigned to a rule, it will not be deleted if you add the NoTransformation column to the items file.
- Once a routing rule is created, it will not be deleted.

# 3

# HL7 Migration Tool

The HL7 Migration Tool allows you to convert lookup tables and transformations developed in other products into InterSystems lookup tables and DTL classes. The HL7 Migration Tool only works if the original transformation was designed to take in an HL7 message and convert it to another HL7 message.

Currently, you can migrate transformations from:

- Cloverleaf
- eGate
- DataGate

In all cases, first make sure you have JRE installed.

## 3.1 Prerequisite: Set Up Java Environment

Before setting up or running the HL7 Migration Tool, you must:

1. Install a current version of a Java Runtime Environment (JRE). Note that it is possible to have multiple versions of a JRE on your system, using the path to the newest one for instructions in this guide.
2. If the JAVA_HOME environment variable doesn’t reflect a valid version, specify the full path for the Java JRE as needed in the instructions below.

## 3.2 Cloverleaf

The Migration Tool allows you to convert Cloverleaf tables and transformations into InterSystems lookup tables and DTL classes. The Migration Tool only works if the Cloverleaf transformation was designed to take in an HL7 message and convert it to another HL7 message.

The Migration Tool is designed to convert basic transformation logic into a DTL class, and is not intended to handle Cloverleaf custom code, loops, and routing rules. Basic TCL commands are converted, but most commands, such as loops, will have to be added to the DTL class after the migration. Code that is not migrated automatically is inserted as comments in the DTL code.

## 3.2.1 Set Up the Migration Tool

Before running the Migration Tool, you must create a working namespace in your InterSystems product and run a setup method.

1. Open an InterSystems Terminal.
2. Enter the following commands to change to the `HSLIB` namespace, create a new namespace for the migration, and change to that new namespace:

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

3. Run the setup tool:

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

4. Answer the prompts of the setup tool.

| Prompt | Enter: |
|---|---|
| Please enter a valid type | Cloverleaf |
| Classname | A new classname used to hold settings defined by the setup tool. This is the class that is used to execute the migration. For example, `MyPkg.MyCloverleaf`. |
| Java ClassPath | The path and file reference to the ANTLR jar file. For example, `c:\Migration\antlr-4.7.2-complete.jar`. |
| Path to java executable | The path and file reference to the java executable on your machine. For example, `c:\java\jdk-13.0.1\bin\java.exe`. |
| Path to java (*.java) files | The path to the *.java files. Be sure to include a trailing `\`. For example, `c:\Migration\Cloverleaf\java\` |

Alternatively, you can pass the parameters to the setup tool like:

```
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")
```

If this command fails, you can view any errors by entering `Do $SYSTEM.Status.DisplayError(status)`.

## 3.2.2 Run Migration Tool

Because the Cloverleaf transformation files can reference data lookup tables, you should convert all of the Cloverleaf lookup tables before converting the transformation files.

### 3.2.2.1 Converting Lookup Tables

Depending on the Cloverleaf environment, you may have multiple lookup tables (`*.tbl`) to convert before converting the transformations. Each lookup table is converted into a persisted lookup table in your InterSystems product. You can run the Migration Tool on a single *.tbl file or a directory containing multiple files. To convert a Cloverleaf lookup table:

1. In the InterSystems Terminal, change to the namespace that you created for the migration. For example:

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

2. If you want to convert multiple *.tbl files at once, put them in a directory and enter:

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

   Where:

   - `MyPkg.MyCloverleaf` is the name of the class you specified while setting up the Migration Tool.
   - `c:\migration\tables\` contains the Cloverleaf files. Be sure to end with a trailing `\`.
   - `CL.Lookup.` is a prefix to the InterSystems tables created with the lookup codes. For example, if a Cloverleaf file is `codes.tbl`, then the InterSystems lookup table would be `CL.Lookup.codes`. This parameter can be any valid package name, but be sure to end with a trailing period ( `.` ).

   To check whether the conversion was successful, enter `write status`. It will display `1` if it was successful.

   To run the Migration Tool on a single Cloverleaf lookup file, enter:

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

### 3.2.2.2 Converting Transformations

Once you have converted the lookup tables used by the HL7 transformations, you are ready to convert the Cloverleaf transformation files into DTL classes. For each Cloverleaf transformation file (*.xlt):

1. In the InterSystems Terminal, change to the namespace that you created for the migration. For example:

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

2. For each Cloverleaf transformation file, enter:

   ```
   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)
   ```

   Where:

   - `MyPkg.MyCloverleaf` is the name of the class you specified while setting up the Migration Tool.
   - `c:\migration\transforms\cloverleaf_sample.xlt` is the Cloverleaf transformation file.
   - `/ClassName` specifies the name of the new DTL class created by the Migration Tool.
   - `/SourceDocType` is the schema and DocType structure of the incoming HL7 message. To view available schemas and DocType structures, open the Management Portal and navigate to **Interoperability** > **Interoperate** > **HL7 v2.x** > **HL7 v2.x Schema Structures**. If you need to add the schema in your InterSystems product, the Message Analyzer can help.
   - `/TargetDocType` is the target schema and DocType structure. The `SourceDocType` is being transformed into the `TargetDocType`.
   - `/TableGroupName` is the name of the lookup table prefix specified while migrating the Cloverleaf lookup table files. If the transformations did not use lookup tables, you can omit this parameter.
   - `/Reprocess` specifies whether to overwrite an existing DTL class. `/Reprocess=1` overwrites the class.
   - `.tOutput` captures information generated by the Migration Tool.

You can check for errors by entering `write status`. If the conversion was successfully, a `1` appears in the Terminal.

By default, the new DTL is compiled after being created by the Migration Tool. To prevent the DTL from being compiled, specify `/BuildDTL=0` as a parameter of the `CodeWalk` method.

# 3.3 eGate

The Migration Tool allows you to convert eGate tables and transformations into InterSystems lookup tables and DTL classes. The Migration Tool only works if the eGate transformation was designed to take in an HL7 message and convert it to another HL7 message.

The Migration Tool is designed to convert basic transformation logic into a DTL class, and is not intended to handle custom code, loops, and routing rules. Code that is not migrated automatically is inserted as comments in the DTL code.

## 3.3.1 Set Up the Migration Tool

Before running the Migration Tool, you must create a working namespace in your InterSystems product and run a setup method.

1. Open an InterSystems Terminal.
2. Enter the following commands to change to the `HSLIB` namespace, create a new namespace for the migration, and change to that new namespace:

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

3. Run the setup tool:

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

4. Answer the prompts of the setup tool.

| **Prompt** | **Enter:** |
|---|---|
| Please enter a valid type | eGate |
| Classname | A new classname used to hold settings defined by the setup tool. This is the class that is used to execute the migration. For example, `MyPkg.MyEGate`. |
| Java ClassPath | The path and file reference to the ANTLR jar file. For example, `c:\Migration\antlr-4.7.2-complete.jar`. |
| Path to java executable | The path and file reference to the java executable on your machine. For example, `c:\java\jdk-13.0.1\bin\java.exe`. |
| Path to java (*.java) files | The path to the *.java files. Be sure to include a trailing `\`. For example, `c:\Migration\eGate\java\` |

Alternatively, you can pass the parameters to the setup tool like:

```
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")
```

If this command fails, you can view any errors by entering `Do $SYSTEM.Status.DisplayError(status)`.

## 3.3.2 Run Migration Tool

Because the eGate transformation files can reference data lookup tables, you should convert all of the eGate lookup tables before converting the transformation files.

### 3.3.2.1 Converting Lookup Tables

Depending on the eGate environment, you may have multiple lookup tables to convert before converting the transformations. Each lookup table is converted into a persisted lookup table in your InterSystems product. You can run the Migration Tool on a single file or a directory containing multiple files. To convert an eGate lookup table:

1. In the InterSystems Terminal, change to the namespace that you created for the migration. For example:

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

2. If you want to convert multiple files at once, put them in a directory and enter:

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

   Where:

   - `MyPkg.MyEGate` is the name of the class you specified while setting up the Migration Tool.
   - `c:\migration\tables\` contains the eGate lookup table files. Be sure to end with a trailing `\`.
   - `eGate.Lookup.` is a prefix to the InterSystems tables created with the lookup codes. For example, if a eGate file is `codes.txt`, then the InterSystems lookup table would be `eGate.Lookup.codes`. This parameter can be any valid package name, but be sure to end with a trailing period ( `.` ).
   - `*.txt` specifies the file extension of the eGate lookup table files.

   To check whether the conversion was successful, enter `write status`. It will display `1` if it was successful.

   To run the Migration Tool on a single eGate lookup file, enter:

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

### 3.3.2.2 Converting Transformations

Once you have converted the lookup tables used by the HL7 transformations, you are ready to convert the eGate transformation files into DTL classes. For each eGate transformation file (*.tsc):

1. In the InterSystems Terminal, change to the namespace that you created for the migration. For example:

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

2. For each eGate transformation file, enter:

   ```
   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)
   ```

Where:

- `MyPkg.MyEGate` is the name of the class you specified while setting up the Migration Tool.
- `c:\migration\transforms\eGate_sample.tsc` is the eGate transformation file.
- `/ClassName` specifies the name of the new DTL class created by the Migration Tool.
- `/SourceDocType` is the schema and DocType structure of the incoming HL7 message. To view available schemas and DocType structures, open the Management Portal and navigate to **Interoperability** > **Interoperate** > **HL7 v2.x** > **HL7 v2.x Schema Structures**. If you need to add the schema in your InterSystems product, the Message Analyzer can help.
- `/TargetDocType` is the target schema and DocType structure. The `SourceDocType` is being transformed into the `TargetDocType`.
- `/TableGroupName` is the name of the lookup table prefix specified while migrating the eGate lookup table files. If the transformations did not use lookup tables, you can omit this parameter.
- `/Reprocess` specifies whether to overwrite an existing DTL class. `/Reprocess=1` overwrites the class.
- `.tOutput` captures information generated by the Migration Tool.

You can check for errors by entering `write status`. If the conversion was successfully, a `1` appears in the Terminal.

By default, the new DTL is compiled after being created by the Migration Tool. To prevent the DTL from being compiled, specify `/BuildDTL=0` as a parameter of the `CodeWalk` method.

# 3.4 DataGate

The Migration Tool allows you to convert DataGate tables and transformations into InterSystems lookup tables and DTL classes. The Migration Tool only works if the DataGate transformation was designed to take in an HL7 message and convert it to another HL7 message.

The Migration Tool is designed to convert basic transformation logic into a DTL class, and is not intended to handle custom code, loops, and routing rules. Code that is not migrated automatically is inserted as comments in the DTL code.

## 3.4.1 Set Up the Migration Tool

Before running the Migration Tool, you must create a working namespace in your InterSystems product and run a setup method.

1. Open an InterSystems Terminal.
2. Enter the following commands to change to the `HSLIB` namespace, create a new namespace for the migration, and change to that new namespace:

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

3. Run the setup tool:

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

4. Answer the prompts of the setup tool.

| Prompt | Enter: |
|---|---|
| Please enter a valid type | DataGate |
| Classname | A new classname used to hold settings defined by the setup tool. This is the class that is used to execute the migration. For example, `MyPkg.MyDataGate`. |
| Java ClassPath | The path and file reference to the ANTLR jar file. For example, `c:\Migration\antlr-4.7.2-complete.jar`. |
| Path to java executable | The path and file reference to the java executable on your machine. For example, `c:\java\jdk-13.0.1\bin\java.exe`. |
| Path to java (*.java) files | The path to the *.java files. Be sure to include a trailing `\`. For example, `c:\Migration\DataGate\java\` |

Alternatively, you can pass the parameters to the setup tool like:

```
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")
```

If this command fails, you can view any errors by entering `Do $SYSTEM.Status.DisplayError(status)`.

## 3.4.2 Run Migration Tool

Because the DataGate transformation files can reference data lookup tables, you should convert all of the DataGate lookup tables before converting the transformation files.

### 3.4.2.1 Converting Lookup Tables

Depending on the DataGate environment, you may have multiple lookup tables to convert before converting the transformations. Each lookup table is converted into a persisted lookup table in your InterSystems product. You can run the Migration Tool on a single file or a directory containing multiple files. To convert a DataGate lookup table:

1. In the InterSystems Terminal, change to the namespace that you created for the migration. For example:

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

2. If you want to convert multiple files at once, put them in a directory and enter:

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

   Where:

   - `MyPkg.MyDataGate` is the name of the class you specified while setting up the Migration Tool.
   - `c:\migration\tables\` contains the DataGate lookup table files. Be sure to end with a trailing `\`.
   - `DataGate.Lookup.` is a prefix to the InterSystems tables created with the lookup codes. For example, if a DataGate file is `codes.txt`, then the InterSystems lookup table would be `DataGate.Lookup.codes`. This parameter can be any valid package name, but be sure to end with a trailing period ( `.` ).
   - `*.txt` specifies the file extension of the DataGate lookup table files.

To check whether the conversion was successful, enter `write status`. It will display `1` if it was successful.

To run the Migration Tool on a single DataGate lookup file, enter:

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

### 3.4.2.2 Converting Transformations

Once you have converted the lookup tables used by the HL7 transformations, you are ready to convert the DataGate transformation files into DTL classes. For each DataGate transformation file (*.tsc):

1. In the InterSystems Terminal, change to the namespace that you created for the migration. For example:

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

2. For each DataGate transformation file, enter:

   ```
   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)
   ```

   Where:

   - `MyPkg.MyDataGate` is the name of the class you specified while setting up the Migration Tool.
   - `c:\migration\transforms\DataGate_sample.tsc` is the DataGate transformation file.
   - `/ClassName` specifies the name of the new DTL class created by the Migration Tool.
   - `/SourceDocType` is the schema and DocType structure of the incoming HL7 message. To view available schemas and DocType structures, open the Management Portal and navigate to **Interoperability** > **Interoperate** > **HL7 v2.x** > **HL7 v2.x Schema Structures**. If you need to add the schema in your InterSystems product, the Message Analyzer can help.
   - `/TargetDocType` is the target schema and DocType structure. The `SourceDocType` is being transformed into the `TargetDocType`.
   - `/TableGroupName` is the name of the lookup table prefix specified while migrating the DataGate lookup table files. If the transformations did not use lookup tables, you can omit this parameter.
   - `/Reprocess` specifies whether to overwrite an existing DTL class. `/Reprocess=1` overwrites the class.
   - `.tOutput` captures information generated by the Migration Tool.

   You can check for errors by entering `write status`. If the conversion was successfully, a `1` appears in the Terminal.

By default, the new DTL is compiled after being created by the Migration Tool. To prevent the DTL from being compiled, specify `/BuildDTL=0` as a parameter of the `CodeWalk` method.