Routing X12 Documents in Productions

Version 2024.1
2024-05-16

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

Routing X12 Documents in Productions
InterSystems IRIS Data Platform Version 2024.1 2024-05-16

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

Introduction to X12

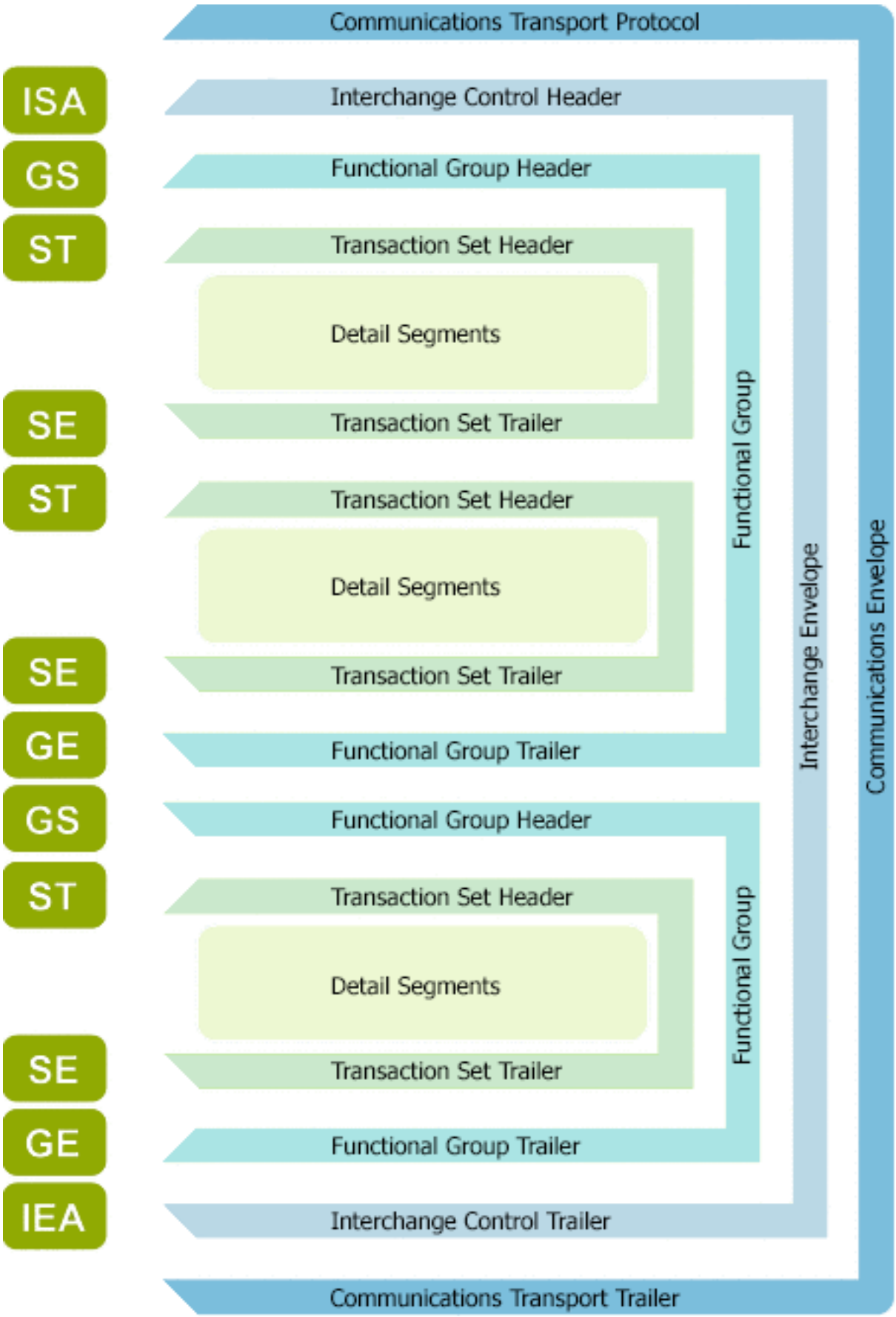
This page briefly introduces the X12 standard and InterSystems IRIS® data platform support for X12.

The American National Standards Institute (ANSI) founded the Accredited Standards Committee (ASC) X12 as a cross-industry forum to build and support electronic data exchange standards, related documents and products intended for worldwide use. Thus, X12 is the ANSI standard for Electronic Data Interchange (EDI). There are more than 300 document types defined. No X12 document type is excluded from InterSystems IRIS support, but most of the focus at InterSystems has been on the documents related to the Health Insurance Portability and Accountability Act (HIPAA).

1.1 The Structure of X12 Envelopes

The rules for X12 envelope structure ensure the integrity of the data and the efficiency of the information exchange. The actual X12 message structure has primary levels that are hierarchical. From highest to the lowest, they are:

- *Interchange Envelope*
- *Functional Group*
- *Transaction Set*


Communications Transport Protocol
ISA
Interchange Control Header
GS
Functional Group Header
ST
Transaction Set Header
Detail Segments
SE
Transaction Set Trailer
ST
Transaction Set Header
Detail Segments
SE
Transaction Set Trailer
GE
Functional Group Trailer
GS
Functional Group Header
ST
Transaction Set Header
Detail Segments
SE
Transaction Set Trailer
GE
Functional Group Trailer
IEA
Interchange Control Trailer
Communications Transport Trailer
Functional Group
Functional Group
Interchange Envelope
Communications Envelope

1.2 X12 and HIPAA

Title II of HIPAA, the Administrative Simplification (AS) provisions, requires the establishment of national standards for electronic healthcare transactions and national identifiers for providers, health insurance plans, and employers. The AS provisions also address the security and privacy of health data. The standards are meant to improve the efficiency and effectiveness of the healthcare system in the United States by encouraging the widespread use of electronic data interchange in the US healthcare system.

Among the X12 documents that specifically support HIPAA are:

- 270_X092: Eligibility, Coverage or Benefit Inquiry
- 271_X092: Eligibility, Coverage or Benefit Information
- 276_X093: Healthcare Claim Status Request
- 277_X093: Healthcare Claim Status Notification
- 278_X094Request: Healthcare Service Review (Request)
- 278_X094Response: Healthcare Service Review (Response)
- 820_X061: Payment Order/Remittance Advice
- 834_X095: Benefit Enrollment and Maintenance
- 835_X091: Healthcare Claim Payment/Advice
- 837_X096: Healthcare Claim: Institutional
- 837_X097: Healthcare Claim: Dental
- 837_X098: Healthcare Claim: Professional

InterSystems IRIS comes prepackaged with HIPAA 4010 and HIPAA 5010 Schemas installed. HIPAA 5010 provides additional documents for health insurance exchange including 277CA, 820hix, 837apD, 837apI, 837apP, and 999.

1.3 X12 Schema Distribution Files

The X12 Standards describe in detail the syntax and semantics of documents. These descriptions are available as PDF files from the official publisher, Washington Publishing Company:

http://www.wpc-edi.com

While informative, the documents are not intended to be machine readable. Machine-readable X12 schemas are available as SEF or XSD files. An SEF file encodes the syntax and much of the semantics of an EDI document. For example, not only does it specify the size and types of fields and any code values which may be applicable, but it also defines dependencies between fields and dependencies between segments within documents. The SEF standard was developed by Foresight Corporation to encode any EDI specification (not just X12) and put into the public domain. You can download a printable version of the standard form:

http://www.edidev.com/articles/sef16.pdf

This information is worth examining, as it gives you an idea of the scope and complexity of EDI. Many organizations have encoded various EDI standards and made them available to the public. In addition, the United States Department of Defense has published many SEF files for their acquisition systems.

The information in the XSD files is mostly the same as in the SEF files, with several notable omissions:

- XSD files are also missing segment position, which is a concept that is also present in the Implementation Guides, but has no impact on InterSystems IRIS processing of the documents.
- Furthermore, XSD files are also missing the segment ordinals, which are actually necessary for identifying a particular occurrence of a segment in the validation-style schema representation. However, ordinal values do not appear in the Implementation Guide, and the default is for them to increment by 1 (only appear in the SEF file when it doesn't follow this pattern), so sticking to the default of incrementing the ordinal by 1 between each segment makes it possible to process documents correctly and does not inaccurately represent the schema.
- Finally, XSD files do not include relational conditions.

1.4 InterSystems IRIS Support for X12 Documents

InterSystems IRIS stores X12 documents as a type of InterSystems IRIS object called a virtual document to enable faster processing. Specifically, an X12 document is stored as an EnsLib.EDI.X12.Document where each segment of the X12 document is stored as an EnsLib.EDI.X12.Segment. X12 Interchanges are also stored as an EnsLib.EDI.X12.Document with pointers to the other X12 documents contained within. Segments and Elements of a Transaction Set are reached by using a virtual property path. For example, the identifier code of a given Transaction Set would be accessed by virtual property path “ST:TransactionSetIdentifierCode”. See Parsing the Document for more information.

InterSystems IRIS provides tools so that you can access values in virtual documents for use in data transformations, business rules, and searching and filtering messages. For background information, see *Using Virtual Documents in Productions*.

InterSystems IRIS provides tools so that you can access values in virtual documents for use in data transformations, business rules, and searching and filtering messages. See Using Virtual Documents in Productions.

Note: In the unlikely event that you wish to clone an X12 batched virtual document, call %ConstructClone(1), passing 1 as the value of the *deep* parameter to create a deep clone, ensuring that both the parent object and any child objects are cloned.

1.5 See Also

- Using Virtual Documents in Productions
- X12 Schemas and Available Tools
- Configuring the Production
- Creating an X12 Data Transformation
- Handling X12 Interchanges
- X12 Settings Reference

2

X12 Schemas and Available Tools

This page provides an overview of the InterSystems IRIS® data platform tools that you can use to work with X12 schemas and documents.

2.1 Using the X12 Schema Structures Page

The **Interoperability** > **Interoperate** > **ASC X12** > **ASC X12 Schema Structures** page enables you to import and view X12 schema specifications.

For general information on using this page, see Using the Schema Structures Page

To determine the schema structures, InterSystems IRIS extracts details from the .SETS, .SEGS, .COMS, .ELMS, and .CODES sections of the SEF file that was imported to define the structure of this X12 document. The description of a document is extracted from the .INI section of the SEF file that defines this X12 schema; an example is Healthcare Claim Status Request.

2.1.1 Loading X12 Schemas into InterSystems IRIS

To load an X12 schema into InterSystems IRIS from the Schema Structures page, click **Import** and select your file. You can load either SEF or XSD files.

For suggestions on where to find SEF files (X12 schemas) to import, see Standard Exchange Format (SEF) Files.

2.1.1.1 Loading SEF Files Programmatically

To load X12 schema files programmatically:

1. Start a Terminal session.
2. Change to an interoperability-enabled namespace and issue the following command:

   ```
   Do ##class(EnsLib.EDI.SEF.Compiler).Import(filename)
   ```

   Where *filename* is the full pathname of the schema file.

   This command imports the data from the schema file and make it available as a schema definition within InterSystems IRIS.
3. InterSystems IRIS creates a name for the new schema category from the first piece of the first line in the .INI section of the SEF file. For example, in 837_X097.SEF you might see this line:

   ```
   .INI 837_X098,,004 010 X098,X,X12-4010,Healthcare Claim: Professional
   ```

The extracted schema category would have this name:

```
837_X098
```

Due to the schema naming convention, if you want to edit a schema file to customize it, InterSystems suggests you first change the text in the schema file that provides its category name, so that you can distinguish your version from any other schema file that you also import into InterSystems IRIS.

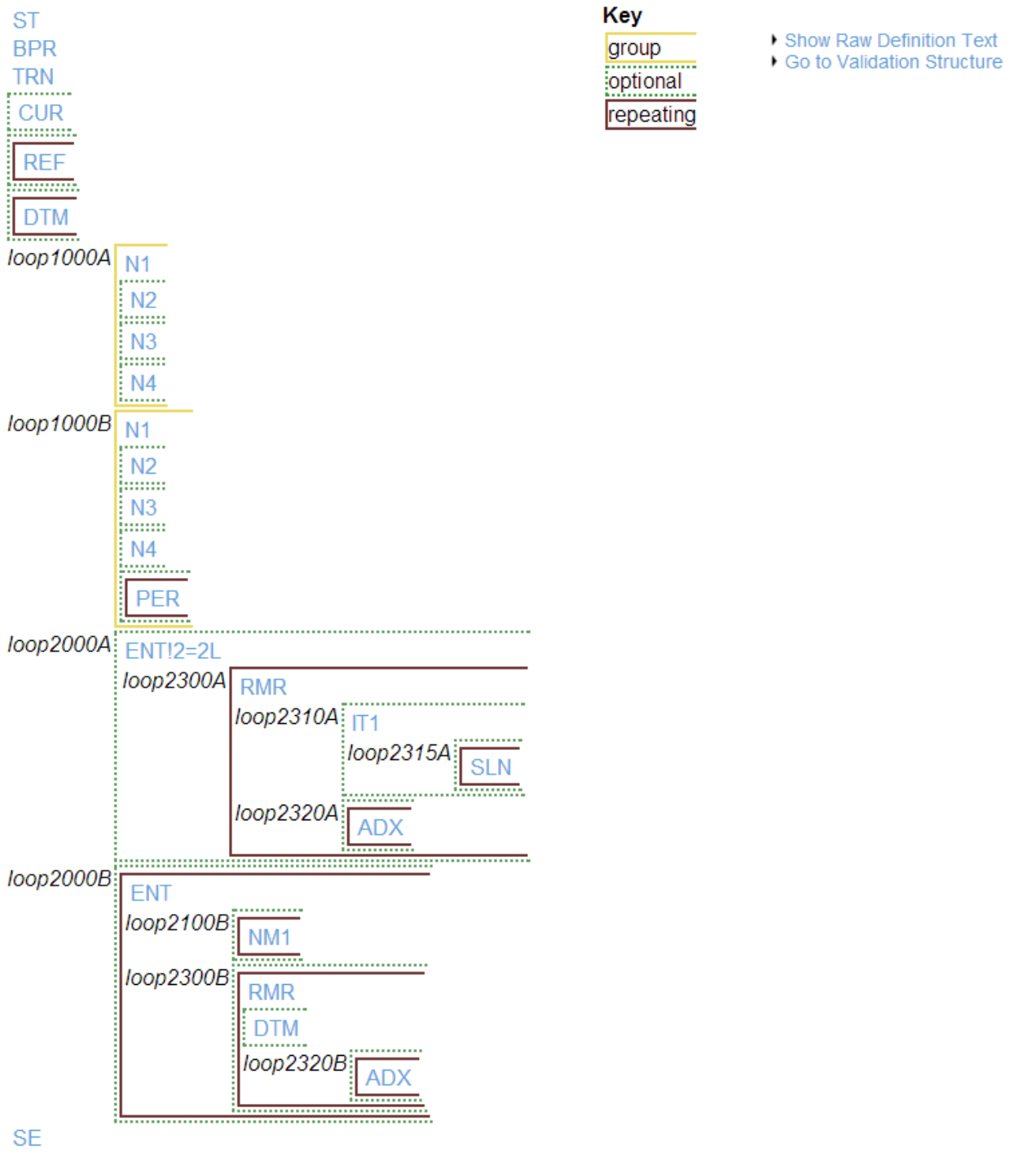
4. A schema file may contain syntax errors. If so, InterSystems IRIS issues an error message and identifies the location of the error in the schema file.

2.1.2 Viewing a Document Structure

To view the internal organization of a document structure, select its name from the **DocType Structures** tab on the **Interoperability** > **Interoperate** > **ASC X12** > **ASC X12 Schema Structures** page. InterSystems IRIS displays the segment structure of the document using the system of visual cues explained below. The following example shows the HIPAA_4010:820 document structure on this page. On the **Interoperability** > **Interoperate** > **ASC X12** > **ASC X12 Schema Structures** page, select **HIPAA_4010** in the left part of the page and select **820** in the right part. This displays the **X12 Document Structure** page.

Schema Category : **HIPAA_4010**
Document Structure (TransactionSet) : **820**
Implementation Guide : **X061A1**

Payment Order/Remittance Advice



2.1.2.1 Layout

- Segments are listed in blue text and link to the relevant segment structure page.

- Loops are listed in black text.
- The segments that comprise the document structure are listed in sequential order, from left to right and top to bottom.

  Note: In accordance with the X12 standard, permitted repetitions of a segment or subloop type may appear in any order prior to the next segment or subloop type in a document. For example, if a schema specifies that eight REF segments must appear prior to a CRC segment, the REF segments may appear in any order. However, they may not be interspersed with other segment types.

- The name of each document segment is displayed: BPR, NM1, DTM, etc. This name indicates the type of segment that exists at this location in the X12 document structure.
- Green dotted lines enclose segments that are optional.
- Brown solid lines enclose segments that, if present, may repeat several times.
- Yellow solid lines enclose segments that are part of a group.
- A segment may be both repeating and optional (see segment NM1 above).
- When you are viewing a segment diagram, if you hover the cursor over a three-letter segment name, a tooltip displays the syntax for referring to this segment in a virtual property path.

2.1.2.2 Raw Definition

To see the document structure in a raw text format, click **Show Raw Definition Text**. The raw definition of the HIPAA_4010:820 document structure is as follows:

Raw Definition

```
ST~BPR~TRN~[~CUR~]~[~{~REF~}~]~[~{~DTM~}~]~(1000A~N1~[~N2~]~[~N
[~N3~]~[~N4~]~[~{~PER~}~]~)~[2000A~ENT!2=2L~{2300A~RMR~[2310A~I
{~ADX~}~]~}~]~[2000B~{~ENT~[2100B~{~NM1~}~]~[2300B~{~RMR~[~DTM
```

Note: Loops in X12 can have custom names. Note in the above raw definition sample for HIPAA_4010:820 that loop titles directly follow open bracket characters "`[,{,(`" unlike segment names, which are separated from bracket characters by a tilde "~". To use a custom loop name, type in your custom name in the place of the default loop name within the schema file prior to importing the schema into InterSystems IRIS.

2.1.2.3 Legacy Document Structure

You can view the old document structure viewer for schemas by clicking **Go to Validation Structure**. The legacy document structure also provides some information that is not available in the default document structure viewer. For example, the legacy document structure can provide code tables, the number of loop repetitions allowed, and syntax notes.

2.1.2.4 Implementation Guide

This code identifies the relevant ASC X12 **Implementation Guide** which can be found at http://www.wpc-edi.com/. The identifier is unique across all document structures and schemas.

2.1.3 Viewing a Segment Structure

To view the structure of a document segment, click on its name in any page similar to the example shown in the previous section. InterSystems IRIS displays a table that lists all the fields in that segment. This is the **Schema Segment Structure** page.

For example, if you click on the PER segment in the HIPAA_4010:276 document structure, InterSystems IRIS displays the following page.

Schema Category: HIPAA_4010
Document Structure (Transaction Set): 276
Segment Structure: **PER**

Path you followed to get to this segment structure: **loop2000A().loop2100A().PER**

HIPAA_4010:PER Administrative Communications Contact

Elem	Description	Property Name	Data Type	Required	Length	Max Repeats	Alternate Description
1	Contact Function Code	ContactFunctionCode	(ID)	O	2	1	
2	Name	Name	(AN)	O	1-60	1	
3	Communication Number Qualifier	CommunicationNumberQualifier	(ID)	O	2	1	
4	Communication Number	CommunicationNumber	(AN)	O	1-80	1	
5	Communication Number Qualifier	CommunicationNumberQualifier2	(ID)	O	2	1	
6	Communication Number	CommunicationNumber2	(AN)	O	1-80	1	
7	Communication Number Qualifier	CommunicationNumberQualifier3	(ID)	O	2	1	
8	Communication Number	CommunicationNumber3	(AN)	O	1-80	1	
9	Contact Inquiry Reference	ContactInquiryReference	(AN)	O	1-20	1	

The columns are as follows:

- **Elem** — the number to use to access the element within the segment (if you prefer numbers).
- **Description** — a short description of the element.
- **Data Type** — a one to two letter symbol representing the element data type. See the table below for details.

SYMBOL	TYPE
Nn	Numeric
R	Decimal
ID	Identifier
AN	String
DT	Date
TM	Time
B	Binary

- **Required** — displays R for required, O for optional.
- **Length** — the number of characters that can be in the element. If only one number is present, it represents the maximum number of characters. If two numbers are separated by a hyphen, it is the range of characters that can be in the associated element (minimum-maximum).
- **Max Repeats** — the maximum number of times the element can repeat.
- **Alternate Description** — a second, longer description of the element.

You can use this information, particularly the **Property Name** column, to build virtual property paths for InterSystems IRIS in the format *segment:elem*. The following are examples of virtual property paths involving simple *elem* values from the

PER segment in the HIPAA_4010:276 document structure. The () shortcut syntax indicates all available instances of a repeating field, whereas (1) indicates the first instance:

```
loop2000A().loop2100A().PER:ContactFunctionCode
loop2000A().loop2100A(1).PER:ContactFunctionCode
loop2000A().loop2100A(2).PER:ContactFunctionCode
loop2000A().loop2100A(x).PER:ContactFunctionCode
loop2000A().loop2100A().PER:Name
```

2.1.4 Viewing a Composite Structure

When you select a name in the **Composite Structure** column, InterSystems IRIS displays all the elements in that data structure. This is the **Composite Structure** page. The column values are identical to those of the previous section.

2.1.5 Choosing a Different Category

It is a feature of the X12 standard that a document structure can differ by X12 version, even when the structure has the same name and number. For example, both X12 HIPAA_4010 and X12 HIPAA_5010 define a document structure called 277, but these definitions contain some differences in segments and segment repetitions. InterSystems IRIS provides the document structure definitions HIPAA_4010:277 and HIPAA_5010:277. The **X12 Document Structure** page makes it easy to see the differences between the two definitions, as the following two figures show.

2.2 Using the X12 Document Viewer Page

InterSystems IRIS provides a Document Viewer page for X12. You can use this page to display, transform, and export X12 documents (either external files or documents from the production message archives).

To access this page, select **Interoperability** > **Interoperate** > **ASC X12** > **ASC X12 Document Viewer**.

2.2.1 Selecting Options

To specify the document to display:

1. For **Document Source**, select **File**, **Message Header ID**, or **Message Body ID**.
2. Specify the document to display:
   - If you selected **File**, use **Browse** to choose a file. For **Document Number in File**, type the number of the document to display.
   - If you selected **Message Header ID** or **Message Body ID**, type the ID of the message header or message body to display.
3. Specify how to parse the document. To do so, select one of the following options for **Document Structure or Schema**:
   - **As received by a Business Service** — Use the schema as assigned by a business service. If you select this, select a business service from the drop-down list.

     This option enables you to determine the DocType to which a particular business service would assign this document.
   - **Use a specific Schema Category/Version** — Choose a document category from the drop-down list.
   - **Use a specific DocType** — Enter the name of an document structure (<MessageStructure>) in the format *category*:*structure*. The parser uses this document structure.
   - **Use content-declared Version:Name** — Use the document structure associated with the document type declared in the document.
   - **Use object's stored DocType** — Use the DocType as declared in the document body object. (This option does not apply to documents loaded from a file.)
   - **None** — Do not use any DocType to parse the document. Instead, display the raw segment without showing any of them as links.

   This option enables you to try interpreting documents from a particular data source as different schema category types to determine which DocType is the right one to use when handling documents from that source. There are a variety of reasons why you might need to do this. For example, you might find when you update an external application that it has changed the actual version of the documents it sends, but has neglected to update the type declaration that it sends in these documents. It is also useful in determining which of the built-in categories to use as a schema base, when a document uses a custom document structure.
4. Optionally click **Transform Document?** and specify the transformation details. See Testing a Transformation.
5. Click **OK**.

2.2.2 Parsing the Document

To parse the document, set the options described above and click **OK**. The Document Viewer displays the following on the right side of the screen:

- A summary of the document which contains following basic information:
  - The Data Transformation applied, if applicable
  - The Document ID
  - The DocType
  - The DocType description, if available
  - The number of segments
  - The Document ID of parent documents, if applicable
- The document data, which has one row for each segment in the document structure. Each row contains:
  - Segment number
  - Segment name, such as ISA or DN1
  - Element contents and separators, as contained in the document

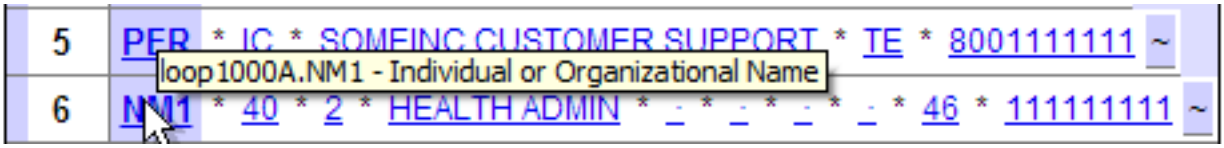
If the document matches the schema you have selected, segments and elements appear as links to the relevant structure page.

Note: Due to the multiplicity of X12 DocType structures that use the same transaction set identifier code (ST:1), InterSystems IRIS checks the additional reference identification numbers ST:3, GS:8, or REF:2 and uses the first implementation guide version number found to uniquely identify document types. For example, examine HIPAA_4010:837P and HIPAA_4010:837D. Both documents have a transaction set identifier of 837, however each has a distinct reference identification number. See Implementation Guide.

2.2.2.1 Displaying the Segment Address

To display a segment address, hover the cursor over a segment name in the shaded column. The tooltip displays the following:

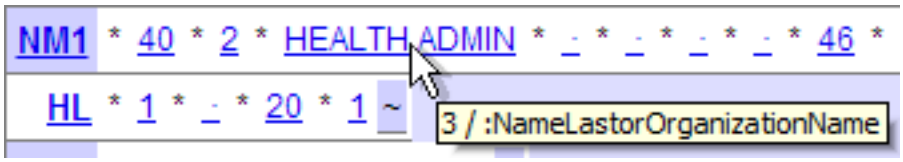
- Segment address to use in a virtual property path
- Descriptive name of this segment



2.2.2.2 Displaying the Element Address

To display the Element address, hover the cursor over a field within the document structure. The tooltip displays the following:

- The *element* address to use in a virtual property path (as a number)
- The *element* address to use in a virtual property path (as a name)

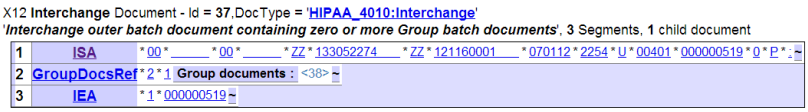


2.2.2.3 Viewing X12 Interchanges

When viewing Transaction Sets nested in Groups and Interchanges the **Interoperability** > **Interoperate** > **ASC X12** > **ASC X12 Document Viewer** page allows you to walk through the document structure one level at a time.

The following display is the result of using the X12 document viewer to view a 4010:Interchange document.

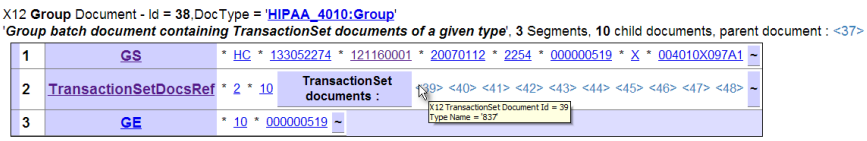
The Document Viewer assigns the group document the identifier <38>.



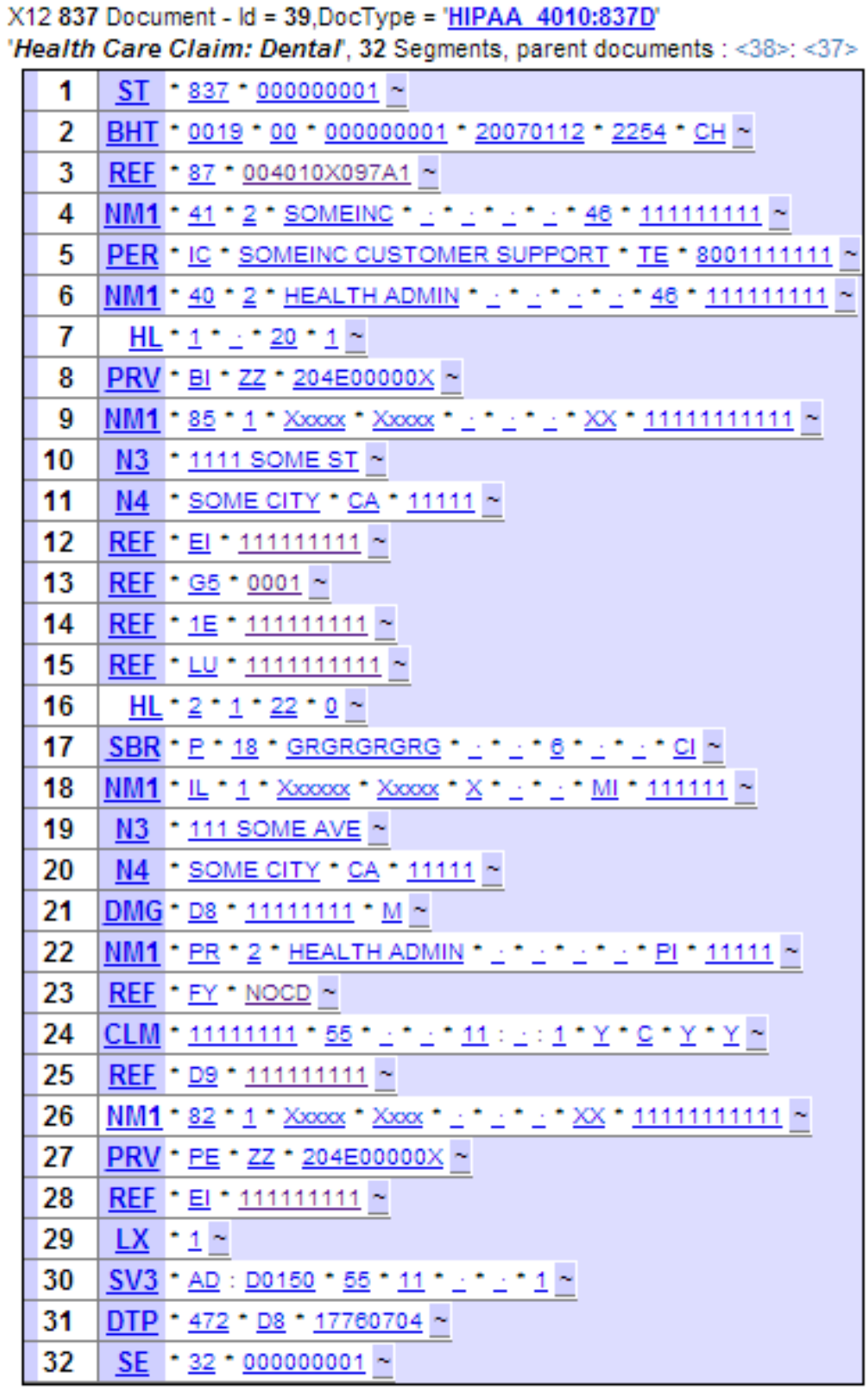
When you click on a group document link in an X12 Interchange document display, a new browser window opens to display the group document. The document Viewer window, with the top-level parent, remains open in the original browser window.

The next display is the result of clicking the link to Group document <38>.

Note: If there are more than 10 Groups in an Interchange or more than 10 Transaction Sets in a Group, the Document Viewer displays links to the first five and last five documents. Between the lists is a text field, into which you can enter any identifier number between the first and last numbers. After you enter a number, click **Other**. A new browser window opens to display the document.



The next display is the result of clicking the Transaction Set <39>. You can return to either the Group or the Interchange by clicking their respective Document ID number links.


X12 837 Document - Id = 39,DocType = 'HIPAA_4010:837D'
'*Health Care Claim: Dental*', 32 Segments, parent documents : <38>: <37>

1	ST * 837 * 000000001 ~
2	BHT * 0019 * 00 * 000000001 * 20070112 * 2254 * CH ~
3	REF * 87 * 004010X097A1 ~
4	NM1 * 41 * 2 * SOMEINC * : * : * : * : * 46 * 111111111 ~
5	PER * IC * SOMEINC CUSTOMER SUPPORT * TE * 8001111111 ~
6	NM1 * 40 * 2 * HEALTH ADMIN * : * : * : * : * 46 * 111111111 ~
7	HL * 1 * : * 20 * 1 ~
8	PRV * BI * ZZ * 204E00000X ~
9	NM1 * 85 * 1 * Xxxxx * Xxxxx * : * : * : * XX * 11111111111 ~
10	N3 * 1111 SOME ST ~
11	N4 * SOME CITY * CA * 11111 ~
12	REF * EI * 111111111 ~
13	REF * G5 * 0001 ~
14	REF * 1E * 111111111 ~
15	REF * LU * 1111111111 ~
16	HL * 2 * 1 * 22 * 0 ~
17	SBR * P * 18 * GRGRGRGRG * : * : * 6 * : * : * CI ~
18	NM1 * IL * 1 * Xxxxxx * Xxxxx * X * : * : * MI * 111111 ~
19	N3 * 111 SOME AVE ~
20	N4 * SOME CITY * CA * 11111 ~
21	DMG * D8 * 11111111 * M ~
22	NM1 * PR * 2 * HEALTH ADMIN * : * : * : * : * PI * 11111 ~
23	REF * FY * NOCD ~
24	CLM * 11111111 * 55 * : * : * 11 : : : 1 * Y * C * Y * Y ~
25	REF * D9 * 111111111 ~
26	NM1 * 82 * 1 * Xxxxx * Xxxx * : * : * : * XX * 11111111111 ~
27	PRV * PE * ZZ * 204E00000X ~
28	REF * EI * 111111111 ~
29	LX * 1 ~
30	SV3 * AD : D0150 * 55 * 11 * : * : * 1 ~
31	DTP * 472 * D8 * 17760704 ~
32	SE * 32 * 000000001 ~

When you are done viewing documents in the batch document hierarchy, you can close all the pop-up browser windows until the top-level parent document remains in the original document Viewer window. From here, you may return to other Management Portal activities.

2.2.3 Testing a Transformation

To test a transformation:

1. Check **Transform Document?**.
2. For **Choose Data Transformation**, select a data transformation.
3. For **Choose Display Option**, select one of the following:
   - **Transformation Result Only** — Display only the transformed document.
   - **Original Message and Result Together** — Display both the original document and the transformed document.
4. Now do either or both of the following:
   - Click **OK** to display the transformed document.
   - Click **Save Result To File?** to save the transformed document to a file. In this case, also specify a path and filename.

     The default directory is the management directory for the active namespace.

2.3 X12 Classes

For reference, this section lists the classes that InterSystems IRIS provides to enable you to work with X12 documents.

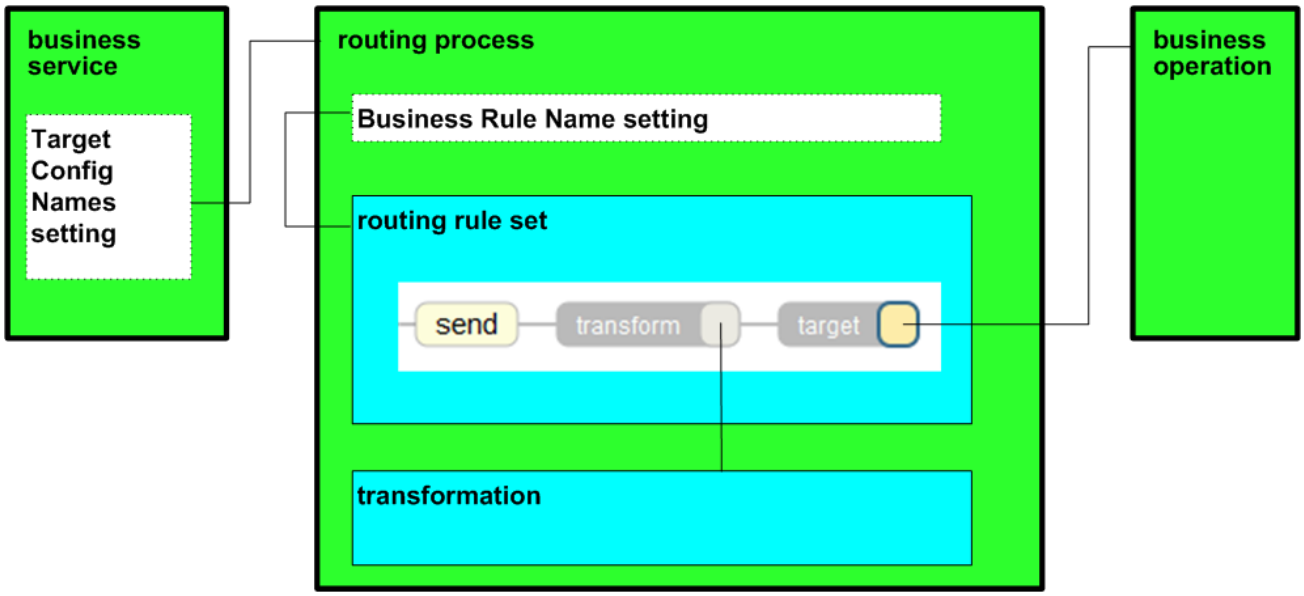
Item	Classes	Notes
X12 business services*	• EnsLib.EDI.X12.Service.FTPService • EnsLib.EDI.X12.Service.FileService • EnsLib.EDI.X12.Service.TCPService • EnsLib.EDI.X12.Service.SOAPService	Each of these X12 business service classes uses a different adapter, as indicated by the class name.
X12 business process*	EnsLib.MsgRouter.VDocRoutingEngine	This class is the standard virtual document business process.
X12 business operations*	• EnsLib.EDI.X12.Operation.FTPOperation • EnsLib.EDI.X12.Operation.FileOperation • EnsLib.EDI.X12.Operation.TCPOperation • EnsLib.EDI.X12.Operation.SOAPOperation	Each of these X12 business operation classes uses a different adapter, as indicated by the class name.
Messages	EnsLib.EDI.Document (automatically used by the business host classes)	This is a specialized message class to carry X12 documents as virtual documents.
Search table	EnsLib.EDI.X12.SearchTable	This is a specialized search table class for X12 documents.

Item	Classes	Notes
Validation and Batch Handling	• EnsLib.EDI.X12.Util.Validator • Ens.X12.FunctionSet	These two classes allow for validation and handling of X12 batches and documents. This includes generating replies and splitting batches in two (one containing children which pass validation and the other containing those which failed validation).

*When you configure a production to work with X12 documents, the Management Portal automatically selects the appropriate business host class.

You can also create and use subclasses of these classes.

The business host classes include configurable targets. The following diagram shows some of them:



For information on other configurable targets, see Reference for Settings.

2.4 Creating Custom X12 Schemas

You can create custom X12 schemas in any of the following ways:

- Create a .x12 file using an XML editor.
- Import an existing schema using the X12 Schemas Management Portal page. Select **Interoperability** > **Interoperate** > **ASC X12** > **ASC X12 Schema Structures** to get to the X12 Schemas page.
- Create an XML file in an IDE and give it an .x12 extension.

You are now ready to edit your file to add and remove XML statements from the <Category> block. The basic editing steps are as follows:

1. Define custom segments using <SegmentStructure> elements.

2. Define custom <MessageStructure> elements that contain the custom segments.
3. Define custom <MessageType> elements that contain the custom message structures.
4. Try viewing the new category definition from the Management Portal. See Portal Tools.

For information on creating custom schema categories, see Creating Custom Schema Categories.

2.5 Defining X12 Search Tables

The X12 search table class, EnsLib.EDI.X12.SearchTable, automatically indexes the X12 document ID, which it gives the name `Identifier`.

If you need more items to search, you can create a subclass. The subclass inherits the `Identifier` property, plus the infrastructure that makes search tables work. For details, see Defining a Search Table Class.

Be sure to perform all tasks in the same namespace that contains your production. Do not use reserved package names; see Reserved Package Names. Also see Overriding the Validation Logic.

2.6 See Also

- Introduction to X12
- Configuring the Production
- Creating an X12 DAta Transformation
- Handling X12 Interchanges
- X12 Settings Reference

3

Configuring the Production

This page describes the process of creating and configuring an X12 production.

Be sure to perform all tasks in the same namespace that contains your production. When you create rule sets and transformations do not use reserved package names; see Reserved Package Names.

Also see Overriding the Validation Logic.

3.1 Creating a New X12 Production

You can add X12 components to an already existing production. However, if you want to create a new production explicitly for handling X12, follow the steps below.

1. In the Management Portal, switch to the appropriate namespace.

   To do so, click the name of the current namespace in the title bar, then click the appropriate namespace in the resulting list.

2. Click **Interoperability**.
3. Click **Configure**.
4. Click **Production** and then click **Go**.

   InterSystems IRIS® then displays the last production you accessed, within the **Production Configuration** page.

5. Click the **Actions** tab on the **Production Settings** menu.
6. Click **New** to invoke the Production Wizard.
7. Enter a **Package Name**, **Production Name**, and **Description**.
8. Choose the **Generic** production type and click **OK**.

InterSystems IRIS creates a blank production from which you can add components such as business services, business processes, and business operations. See the sections below for more details.

Note: As you build your production, it frequently happens that while configuring one component you must enter the name of another component that you have not yet created. *A clear naming convention is essential to avoid confusion.* For suggestions, see Naming Conventions. For rules, see Configuration Names.

3.2 Adding an X12 Business Service

Add one X12 business service for each application or source from which the production receives X12 documents.

To add an X12 business service to a production:

1. Access the Business Service Wizard as usual; see Configuring Productions.
2. Click the **X12 Input** tab.
3. Click one of the following from the **Input type** list:
   - **TCP**
   - **File**
   - **FTP**
   - **SOAP**
4. In the **X12 Service Name** field, type the name of this business service.
5. In the **X12 Service Target** area, select one of the following options:
   - **Create New Router** — InterSystems IRIS adds a business process to the production and configures the business service to use it as a target. You can edit the business process details later.
   - **Choose From List** — In this case, also select an existing business host from the drop-down list.
   - **None for Now** — Do not specify a target for this business service. If you make this selection, ensure that you specify a target later.
6. Click **OK**.

3.3 Adding an X12 Business Process

To add an X12 business process to a production:

1. Access the Business Process Wizard as usual; see Configuring Productions.
2. Click the **X12 Router** tab; the router class defaults to EnsLib.EDI.X12.MsgRouter.RoutingEngine.
3. For **Routing Rule Name**, do one of the following:
   - Select an existing routing rule from the **Routing Rule Name** drop-down list.
   - Select **Auto-Create Rule** and type a rule name into **Routing Rule Name**. In this case, the wizard creates the routing rule class in the same package as the production.

     Later you must edit the routing rule and add your logic to it.
4. For **X12 business process Name**, type the name of this business process.
5. Click **OK**.
6. Ensure that your X12 business service is connected to the new X12 Business Process. To connect the process:
   - Select your X12 business service.

- Click the **Settings** tab and open the **Basic Settings** menu in the menu to the right of the screen.
- Enter the name of the new X12 business process in the **Target Config Names** field.

7. Configure additional settings of the business process, as needed. For details, see Settings for X12 Business Processes.

3.4 Adding an X12 Routing Rule

For general information on defining business rules, see Developing Business Rules.

When you create an X12 routing rule:

- On the **general** tab, **Rule Type** should be **Virtual Document Message Routing Rule**. This choice sets the following options:
  - **Rule Assist Class** should be EnsLib.MsgRouter.VDocRuleAssist
  - **Context Class** should be EnsLib.MsgRouter.VDocRouting Engine
- In the **constraint** for a rule, specify **Message Class** as EnsLib.EDI.X12.Document.

In all other respects, the structure and syntax for both types of rule set are the same.

Note that for X12 routing rules, the rule editor provides a test option where you can paste the raw text of an X12 message.

3.5 Adding an X12 Business Operation

To send X12 messages from a production to a file or application, you must add an X12 business operation. Add an X12 business operation for each output destination.

You might also want to add business operations to handle bad messages (for background, see Business Processes for Virtual Documents).

To add an X12 business operation to a production:

1. Access the Business Operation Wizard as usual; see Configuring Productions.
2. Click the **X12 Output** tab.
3. Click one of the following from the **Output type** list:
   - **TCP**
   - **File**
   - **FTP**
   - **SOAP**
4. For **X12 Operation Name**, type the name of this business operation.
5. Click **OK**.
6. Ensure that the business operation is connected to the relevant business services or business process
   - For a routing rule, enter the name of your X12 business operation in the **Target** field of the routing rule set.

- If your design uses a pass-through interface that simply relays messages from the incoming business service to the outgoing business operation, enter the name of your X12 business operation in the **Target Config Names** field of the X12 business service.

7. Configure additional settings of the business operation, as needed. For details, see Settings for X12 Business Operations.

If you want the production to send data that is not an X12 message, see Defining Business Operations. Also see Connectivity Options.

3.6 See Also

- Introduction to X12
- X12 Schemas and Available Tools
- Creating an X12 Data Transformation
- Handling X12 Interchanges
- X12 Settings Reference

4

Creating an X12 Data Transformation

Your X12 routing rule may need one or more data transformations. For general information on defining Data Transformation Language (DTL) data transformations, see Developing DTL Transformations.

To integrate the DTL data transformation in the production, enter its full package and class name in the **Transform** field of a routing rule.

When you create a DTL data transformation for X12 documents:

- On the **Transform** tab, **Source Class** and **Target Class** should both be EnsLib.EDI.X12.Document.
- **Source Doc Type** should match the schema category name assigned by the business service.
- **Target Doc Type** should be the name of the target schema category. This must match a schema category name that you have loaded into InterSystems IRIS.

4.1 The Common X12 DTL Use Case

The DTL editor facilitates creation of data transformations for the common X12 usage case where:

- You can have more than one functional group.
- Normally, there is only one type of transaction set used within a group.

In general each DTL has a single schema for the input message and a single schema for the output message. This works for messages where a single schema describes the message, but in X12 there are different schemas for the three different levels:

- Schema for the Interchange Envelope
- Schema for the Function Group
- Schema for the Transaction Set

In DTL, you handle this by having a subtransform for each level, but the DTL X12 support allows you to specify a schema for each level in the hierarchy. If you are dealing with a more complex X12 message, such as one with multiple types of transaction sets, you would have to use subtransforms to handle this.

4.2 Steps for Creating an X12 DTL

This section describes how to create a data transformation that copies the content of a source X12 document to a target X12 document. The example uses the HIPAA 5010 834 document structure for both the source and target documents. You use similar procedures for other formats.

Because of the hierarchical structure of X12 messages, as described in The Structure of X12 Envelopes, you cannot simply set the value of the **Create** field on the **Transform** tab to **Copy**. Instead, you build a data transformation that copies each level of the message structure. The class Ens.X12.FunctionSet, which is a sub class of Ens.Rule.FunctionSet, provides the following class methods to help with this task:

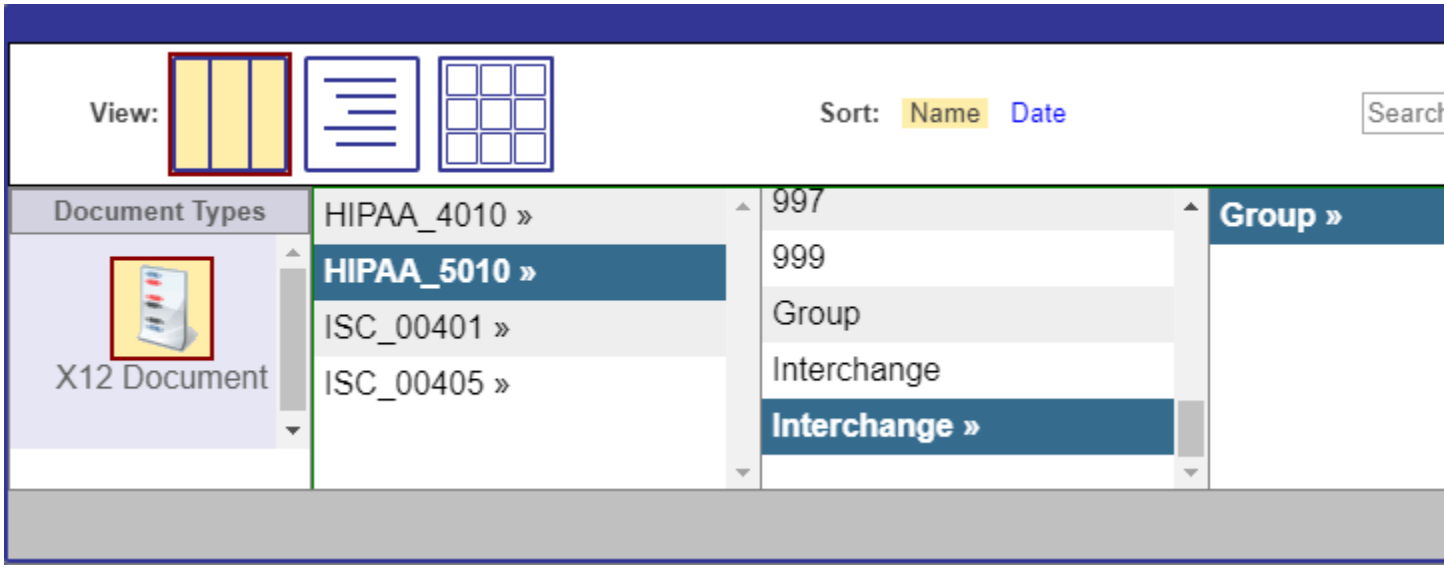
- **X12NewFunctionalGroup()**
- **X12NewTransactionSet()**
- **X12CompleteTrailerAndReturnClear()**
- **X12SaveDocumentAndReturnClear()**

4.2.1 Creating Source and Target Classes

First, create source and target data transformation classes for the appropriate document type:

1. From the **Data Transformation Builder** page, click **New** to open the **Data Transformation Wizard**.
2. In the **Package** field, enter the name of the package the new class belongs to, or click the arrow to select a package in the current namespace.
3. In the **Name** field, enter a name for the new data transformation class.
4. In the **Description** field, enter a description for the data transformation.
5. Select **X12** for both the **Source Type** and **Target Type**, this action automatically enters EnsLib.EDI.X12.Document as **Source Class** and **Target Class**.
6. To select a **Source Document Type**, click on the looking glass icon.
   - In the resulting dialog box, select the appropriate schema category, for example, HIPAA_5010.
   - In the column to the right, select **Interchange>>** .
   - In the next column to the right, select **Group>>**. This is the schema for the functional groups.
   - In the next column to the right, select the name representing the document type, in this example, 834. This is the schema for the transaction sets.
7. These actions give you a **Source Document Type** of `HIPAA_5010:Interchange:Group:834` and defaults to the same type for **Target Document Type**.
8. Click **OK** to create the new source and target classes.

The following image shows the process of selecting the **Source Document Type**.



4.2.2 Copying the Interchange Control Header

You may want to copy the content of the Interchange Control Header (ISA) from source to target. You can use the drag-and-drop technique described in Copying a Value from a Source Property to a Target Property in Developing DTL Transformations to create the **set** actions. You may also want to gather these actions together in a **group** action.

4.2.3 Copying Functional Groups

1. Create a new `group` action by clicking **Add Action** and selecting **group**.
2. Select the **FunctionalGroups()** node in the diagram.
3. Create a new `for each` action by clicking **Add Action** and selecting **for each**. Having selected **FunctionalGroups()** provides a default value for the **Property** field.
4. On the **Action** tab, the **Property** field for the new `for each` action has the default value `source.FunctionalGroups()`, which instructs the `for each` to iterate through the functional groups in the source class.
5. The **Key** field contains the variable used to iterate through the loop. It has the default value `k1`. Change it to a more memorable variable name, such as `fgKey`, for functional group key.
6. Create a new `set` action in the `for each` loop by clicking **Add Action** and selecting **set**. In the **Property** field, add `target.FunctionalGroups(fgKey)`.
7. To edit the **Value** field, click on the magnifying glass icon to launch the value editor.
8. In the value editor dialog box, click the arrow to the right of the **Function** field to open the drop down list. Select the function `X12NewFunctionalGroup()`.
9. In the **InterchangeParent** field, type `target`, and in the **DocType** field, type `"Group"` and click **OK**.
10. This action adds `##class(Ens.X12.FunctionSet).X12NewFunctionalGroup(target,"Group")` to the **Value** field. The entire `set` action uses the **X12NewFunctionalGroup()** method to create a functional group in the target class.

The next step is to copy the transaction sets.

4.2.4 Copying Transaction Sets

1. Inside the `for each` action from the previous section, create a new `group` action by clicking **Add Action** and selecting **group**.
2. Select the **TransactionSets()** node in the diagram.
3. Create a new `for each` action by clicking **Add Action** and selecting **for each**. Having selected **TransactionSets()** provides a default value for the **Property** field.
4. On the **Action** tab, the **Property** field for the new `for each` action has the value `source.FunctionalGroups(fgKey).TransactionSets()`, which instructs the `for each` to iterate through the transaction sets in the source class.
5. The **Key** field contains the variable used to iterate through the loop. It has the default value `k2`. Change it to a more memorable variable name, such as `tsKey`, for transaction set key.
6. Create a new `set` action by clicking **Add Action** and selecting **set**. In the **Property** field, add `target.FunctionalGroups(fgKey).TransactionSets(tsKey)`.
7. To edit the **Value** field, click on the magnifying glass icon to launch the value editor.
8. In the value editor dialog box, click the arrow to the right of the **Function** field to open the drop down list. Select the function `X12NewTransactionSet()`.
9. In the **GroupParent** field, type `target.FunctionalGroups(fgKey)`, and in the **DocType** field, type `"834"` and click **OK**.
10. This action adds `##class(Ens.X12.FunctionSet).X12NewTransactionSet(target.FunctionalGroups(fgKey),"834")` to the **Value** field. The entire `set` action uses the **X12NewTransactionSet()** method to create a transaction set in the target class.

4.2.5 Complete the Functional Group and Transaction Set Loops

In the preceding two sections, you created two loops, one that iterates over the functional groups in the source and creates a corresponding functional group in the target, and a second nested loop that creates the required transaction sets in each of the functional groups. Now you need to create the trailer segment for each transaction set, and for each of the functional groups that contains them.

To create the transaction set trailer segment:

1. In a location just before the end of the loop that creates the transaction sets, create a new `set` action by clicking **Add Action** and selecting **set**. In the **Property** field, add `target.FunctionalGroups(fgKey).TransactionSets(tsKey)`.
2. To edit the **Value** field, click on the magnifying glass icon to launch the value editor.
3. In the value editor dialog box, click the arrow to the right of the **Function** field to open the drop down list. Select the function `X12CompleteTrailerAndReturnClear()`.
4. In the **X12Document** field, type `target.FunctionalGroups(fgKey).TransactionSets(tsKey)`, and in the **Save** field, type 1 and click **OK**.
5. This action adds `##class(Ens.X12.FunctionSet).X12CompleteTrailerAndReturnClear(target.FunctionalGroups(fgKey).TransactionSets(tsKey),1)` to the **Value** field. The entire `set` action uses the **X12CompleteTrailerAndReturnClear()** method to create the transaction set trailer for the current transaction set.

To create the functional group trailer segment:

1. In a location just before the end of the loop that creates the functional groups, create a new `set` action by clicking **Add Action** and selecting **set**. In the **Property** field, add `target.FunctionalGroups(fgKey)`.
2. To edit the **Value** field, click on the magnifying glass icon to launch the value editor.
3. In the value editor dialog box, click the arrow to the right of the **Function** field to open the drop down list. Select the function `X12CompleteTrailerAndReturnClear()`.
4. In the **X12Document** field, type `target.FunctionalGroups(fgKey)`, and in the **Save** field, type 1 and click **OK**.
5. This action adds `##class(Ens.X12.FunctionSet).X12CompleteTrailerAndReturnClear(target.FunctionalGroups(fgKey),1))` to the **Value** field. The entire `set` action uses the **X12CompleteTrailerAndReturnClear()** method to create the functional group trailer for the current functional group.

Finally, create the IEA segment:

1. In a location just after the end of the loop that creates the functional groups, create a new `set` action by clicking **Add Action** and selecting **set**. In the **Property** field, add `temp`.
2. To edit the **Value** field, click on the magnifying glass icon to launch the value editor.
3. In the value editor dialog box, click the arrow to the right of the **Function** field to open the drop down list. Select the function `X12CompleteTrailerAndReturnClear()`.
4. In the **X12Document** field, type `target`, and in the **Save** field, type 0 and click **OK**.
5. This action adds `##class(Ens.X12.FunctionSet).X12CompleteTrailerAndReturnClear(target,0))` to the **Value** field. The entire `set` action uses the **X12CompleteTrailerAndReturnClear()** method to create the IEA segment.

4.3 Data Transformations for Whole Batch

To transform whole batch documents at the group or transaction set level, you must iterate through each group and each transaction set within each group. To do so, follow the steps below.

4.3.1 Loop over the Groups in each Interchange

1. Create a new `for each` action by clicking **Add Action** and selecting **for each**.
2. Input **source.{GroupDocsRef}** into the **Property** field.
3. Input `Group` into the **Key** field. `Group` is the actual group-level document object.
4. Add a subtransformation on each Group. The value passed into the subtransformation should be the Group (the key from the `for each`) and then the Property is a new Group, which could be named GroupOut.
5. The source and target of the main Transformation are Interchange documents. In this subtransformation, the Group documents are the source and target, thus enabling you to easily manipulate the Group segments through the drag-and-drop capabilities of DTL, and also enabling you to easily loop over the individual Transaction Sets.

4.3.2 Loop over the Transaction Sets in each Group

1. Within the Group subtransformation, create a new `for each` action by clicking **Add Action** and selecting **for each**.

2. Then perform another subtransformation on each of the Transaction Sets. We pass in the Transaction Set object that was the `for each` Key. The returned property is also a Transaction Set. This subtransformation has Transaction Set DocTypes for its Source and Target, for example HIPAA_5010:834 or HIPAA_5010:837P.
3. If there is some possibility of the batch having Transaction Sets of varying DocTypes, you could add conditions on the DocType to determine which Transaction Set subtransformation to perform. This scenario could be a reason to use this approach rather than that described in the previous section.

In each of the Group transformation and the Interchange transformation, after the subtransformation, but before the close of the ForEach, we need to add the returned child object to the parent target document. Doing so will require a custom function. This custom function can be something like:

```
/// Adds a child X12 document <var>pChild</var>, (a Transaction Set or a Group)
as the child of a Group or Interchange

/// <var>pParent</var>'s DocsRef segment. <var>pDocsRefSeg</var>
is the segment name for the DocsRef segment

/// ("TransactionSetDocsRef" for a Group or "GroupDocsRef" for an Interchange)
Returns a status code.

ClassMethod AddChildToDocsRef(pChild As EnsLib.EDI.X12.Document,
 pParent As EnsLib.EDI.X12.Document, pDocsRefSeg As %String) As %Status
{
    Set tSC = $$$OK
    Set tSC = pParent.BuildMap() Quit:$$$ISERR(tSC) tSC
    Set index = pParent.GetSegmentIndex(pDocsRefSeg,.tSC) Quit:$$$ISERR(tSC) tSC
    Set tSC = pParent.AddChild(index,pChild) Quit:$$$ISERR(tSC) tSC
    Set tSC = pParent.SetValueAt(1+pParent.GetValueAt((index+1)_":1"),(index+1)_":1")
     Quit:$$$ISERR(tSC) tSC
    Set tSC = pParent.%Save()
    Quit tSC
}
```

4.4 Testing an X12 Data Transformation

The **Tools** tab in the DTL editor provides a **Test** button that lets you quickly test a data transformation. Click **Test**, paste X12 text into the **Input Message** field, and the **Test** button in the **Test Transform** dialog.

You can also use the X12 Document Viewer Page to test your transformations, as described in Available Tools. The document viewer allows you to run a DTL against different X12 data sources.

Note: Be sure your data transformation reflects your Batch Handling settings.

4.5 See Also

- Introduction to X12
- X12 Schemas and Available Tools
- Configuring the Production
- Handling X12 Interchanges
- X12 Settings Reference

5

Handling X12 Interchanges

As X12 documents are sent within Interchanges you need to choose how InterSystems IRIS® should handle incoming documents. Especially if you need to access documents at the Group or Transaction Set level. This page describes different scenarios for handling X12 Interchange.

5.1 X12 Batch Handling

Batch Handling settings determine how InterSystems IRIS sends and receives documents. The choices you make here affect how your Business Processes should be configured.

5.1.1 Receiving Batch Documents

X12 business services have the Batch Handling configuration setting, which determines how to process incoming batch documents. The options are:

- `Whole Batch` — Do not process child documents individually; accumulate and send the whole batch as one composite document.
- `Single-Session Batch` — Forward all documents in the Interchange as part of a single session, including final parent document objects containing batch and group header and trailer segments.
- `Multi-Session Batch` — Forward each document in the Interchange in its own session, followed by final parent document objects containing the batch and group header and trailer segments.
- `Individual` — Forward each child document in the batch in its own session; do not forward parent batch document objects.

Note: If you select **Whole Batch**, any transformations of individual Transaction Sets have to account for the Interchange and Group levels. See Configuring Business Processes for Whole Batch below.

5.1.2 Sending Batch Documents

On the outgoing side, X12 File and FTP business operations have the Auto Batch Parent Segs configuration setting. When **Auto Batch Parent Segs** is False (the default) the business operation outputs child documents, but does not output the batch headers and trailers. When **Auto Batch Parent Segs** is True, while outputting a message that has a batch parent, the business operation outputs the batch headers first, then the child documents, then follows up with the batch trailers when triggered by the final batch header message or by a file name change.

In situations where the trailing segments might not be written correctly, such as when the business service used **Batch Handling** = **Individual**, you can use the Auto Batch Completion Timeout setting for File and FTP business operations to ensure that trailing segments are written.

5.1.3 Batch Modes

The combination of **Batch Handling** and **Auto Batch Parent Segs** enables the following modes of operation for X12 batch documents:

Batch Handling	Auto Batch Parent Segs	Results
`Whole Batch`	(any)	Business service sends only the parent document; all child documents are referenced to it but not sent individually. Operation outputs entire batch at one time when it receives the parent document.
`Single-Session` or `Multi-Session`	`True`	Service sends each child document as it receives and parses it, followed by the parent document when all children have been sent. The business operation outputs parent headers when it receives the first child document, then finishes up with trailers when it receives the parent document object. Trailer segments automatically contain the correct child count values.
`Single-Session` or `Multi-Session`	`False`	This results in double output: the business operation sends out each child document individually, followed by the parent document containing each child document (again).
`Individual`	`False`	Business service forwards each child document in the batch in its own session and does not forward objects representing the batch headers and trailers. On the outgoing side, the business operation does the same.
`Individual`	`True`	Business service forwards each child document in the batch in its own session and does not forward objects representing the batch headers and trailers. On the outgoing side, the business operation does the same.

5.2 Configuring Business Processes for Whole Batch

If you choose to have the business service process incoming X12 documents in Whole Batch mode you need to configure the business process accordingly.

5.2.1 Routing Rules for Whole Batch

To properly route documents in Whole Batch mode, set the routing rule to accept Interchanges. Send to the relevant data transformation if necessary.

5.2.1.1 Create a new routing rule

1. Ensure you are in the proper namespace and navigate to the Rule Editor page.

2. Click **New**.
3. Follow the instructions in Adding an X12 Routing Rule.

5.2.1.2 Edit the constraint

1. Double click on the **constraint** rule item.
2. Set the **Source** to your business service.
3. Set the **Message Class** to `EnsLib.EDI.X12.Document`.
4. Set the **Schema Category** as appropriate for your production.
5. Choose `Interchange` from the **Document Name** selection.

For more information, see Using the Rule Constraint Editor.

5.2.1.3 Define the condition

Choose any conditional statement relevant to your production. If you always want the action to be applied, click on the **condition** rule item and type “1”.

5.2.1.4 Add a Send action

1. Click on the **when** rule item.
2. Click **Send** to add a send action.
3. Double click on the **transform** rule item and select the appropriate data transformation.
4. Double click on the **target** rule item and select the appropriate business operation from your production.

For more information, see Selecting the Transformation and Target of a Send Action.

5.3 Configuring Business Processes for Other Batch Settings

To transform X12 documents within productions with Single Session, Multi Session, or Individual mode you must create a separate data transformation for each type of X12 document the production encounters. For more information, see Developing DTL Transformations.

5.3.1 Routing Rules for Other Batch Settings

To properly route documents in Single Session, Multi Session, or Individual mode, create a routing rule set with a unique routing rule for each type of X12 document the production encounters.

To create new rules within the rule set:

1. Click on **ruleSet**
2. Click **rule** to add a rule
3. Follow the instructions set in Routing Rules for Whole Batch
4. When editing the constraint, be sure to set the **Document Name** to the appropriate X12 document type.

5. When adding a data transformation, be sure to use a transformation written for the appropriate X12 document type.

5.3.2 Data Transformations for Other Batch Settings

To transform X12 documents within productions with Single Session, Multi Session, or Individual mode you must create a separate data transformation for each type of X12 document the production encounters. For more information, see Creating an X12 Data Transformation and Developing DTL Transformations.

5.4 See Also

- Developing DTL Transformations
- Introduction to X12
- X12 Schemas and Available Tools
- Configuring the Production
- Creating an X12 Data Transformation
- X12 Settings Reference

X12 Settings Reference

This section provides reference information for X12 business hosts.

For information on settings for the business process (EnsLib.MsgRouter.VDocRoutingEngine), see Settings of a Virtual Document Routing Process.

Settings for X12 Business Services

Provides reference information for settings of an X12 business service.

Summary

X12 business services have the following settings:

Group	Settings	See
Basic Settings	Target Config Names, Doc Schema Category, Batch Handling	Settings for Business Services
Acknowledgement	Reply Target Config Names (File), Validation Mode, Validation, SNIP Level, Batch Error Action, Batch Reply Type, AddNackErrText, Reply Mode, BadMessageHandler	*sections in this topic*
Connection Settings	Tolerate Newlines	*sections in this topic*
Additional Settings	Search Table Class	Settings for Business Services
	Reply Target Config Names (FTP)	*section in this topic*
	Local Application ID, Default Char Encoding,	*sections in this topic*

The remaining settings are either common to all business services or are determined by the type of adapter. For information, see the following sections:

- Settings for All Business Services
- Settings for the File Inbound Adapter
- Settings for the FTP Inbound Adapter

  EnsLib.EDI.X12.Adapter.TCPInboundAdapter has Job Per Connection set to False, which is usually appropriate for X12.
- Settings for the SOAP Inbound Adapter

Add Nack Error Text

Add extra error-text field to TA1 & 997/999 AKx segments when generating error-status reply messages; otherwise do not embed internal error state information in the Acknowledgement messages.

Bad Message Handler

If a document fails validation, it will be sent to the **Bad Message Handler** specified by this setting. If the document is part of a batch, the Batch Error Action determines how the batch is handled. This setting acts in combination with the Validation Mode, Validation, SNIP Level, and Batch Error Action

See Defining Bad Message Handlers.

Batch Error Action

Specifies what the service does when it detects a validation error in a batch Interchange document. The options are as follows:

`Reject With All Errors`

If the service encounters an error in a document within the batch, it validates the remainder of the batch, and then rejects the whole batch.

Additionally, if the Batch Reply Type allows, the reply document enumerates all the errors and shows an acknowledgement code of either A (accepted) or R (rejected) in each Functional Group Response Trailer (AK9) segment and Transaction Set Response Trailer (AK5) segment, as appropriate.

The service does not forward the batch or any part of the batch to the target business host.

`Reject On First Error`

If the service encounters an error in a document within the batch, it rejects the whole batch immediately. The service does not check for additional errors or parse the remainder of the batch.

Additionally, if the Batch Reply Type allows, the reply document enumerates the error and shows an acknowledgement code of either A (accepted) or R (rejected) in each validated Functional Group Response Trailer (AK9) segment and Transaction Set Response Trailer (AK5) segment, as appropriate. No Transaction Set Response Header (AK2) segments appear after the AK5 segment marked with R, which corresponds to the transaction set that generated the error.

The service does not forward the batch or any part of the batch to the target business host.

`Reject Individual Errors`

The service forwards any documents in the batch that pass validation to the target business host.

The service does not send transaction sets that include errors to the target business host and removes invalid transaction sets from any functional groups it sends. If all the transaction sets in a functional group include errors, the service does not send the functional group. Similarly, if all the transaction sets in all the functional groups in an interchange include errors, the service does not send the interchange.

Additionally, if the Batch Reply Type allows, the reply document enumerates the errors, shows an acknowledgement code of either A (accepted), P (partially accepted), or R (rejected) in each Functional Group Response Trailer (AK9) segment, and shows an acknowledgement code of either A or R in each Transaction Set Response Trailer (AK5) segment, as appropriate.

`Reject Individual Errors` is the default value.

`Accept With Errors`

The service forwards all documents in the batch.

Additionally, if the Batch Reply Type allows, the reply document enumerates the errors and shows an acknowledgement code of either A (accepted) or E (accepted but errors were noted) in each Functional Group Response Trailer (AK9) segment and Transaction Set Response Trailer (AK5) segment, as appropriate.

Note: If Reply Mode is `Application` and Batch Handling is not `Individual`, InterSystems IRIS® may forward some of the documents in a batch before rejecting the whole batch upon encountering an error.

When the service does not forward a document due to a validation error, the document is deleted. Additionally, if the Batch Handling value is `Individual`, persisted parent documents are deleted if none of their corresponding children are forwarded.

Batch Handling

X12 Transaction Set documents are often packaged in a batch document called an Interchange, which contains nested sub-batches called Functional Groups. The **Batch Handling** setting specifies how InterSystems IRIS treats received document batches. The options are as follows:

- `Whole Batch` — Do not process child documents individually; accumulate and send the whole batch as one composite document.
- `Single-Session Batch` — Forward all documents in the batch together in one session; the session includes objects representing the parent document header and trailer segments. This is the default.

- `Multi-Session Batch` — Forward each document in the batch in its own session, including the objects representing the batch header and trailer segments.
- `Individual` — Forward each child document in the batch in its own session; do not forward objects representing the parent batch document's header and trailer segments.

Batch Reply Type

Specifies the type of batch reply to create for an Interchange batch that has been received. The following table lists the possible choices:

Value	Meaning
None	Do not generate a batch reply. If an error occurs, do not create any immediate notification reply to the sender.
All	Generate a reply Interchange containing a reply notification for every Transaction Set received in the Interchange.
All+TA1	Generate a reply Interchange containing a TA1 segment that indicates acceptance or error status for the entire Interchange, and a reply notification for every Transaction Set received in the Interchange.
All+ISA14TA1	Generate a reply Interchange containing a TA1 segment and a response for each Transaction Set only if a `1` appears in the Acknowledgement Requested (ISA:14) field of the Interchange Control Header (ISA) segment or if an error appears in the Interchange Envelope. Alternatively, generate an Implementation Acknowledgement (999) document with a response for each Transaction Set.
Errors	Whether or not errors are found, generate a reply Interchange. If no errors are found, generate an empty reply Interchange. If errors are found, generate an Interchange that contains reply notifications only for Transaction Sets in which errors are detected. This is the default setting if no choice is specified.
OnlyIfErrors	If errors are found, generate a reply Interchange that contains reply notifications only for Transaction Sets in which errors are detected.
Successes	Whether or not errors are found, generate a reply Interchange. If errors are found for every Transaction Set, generate an empty reply Interchange. Otherwise, generate a reply Interchange that contains reply notifications only for Transaction Sets in which no errors are detected (successes).
TA1	Generate a reply Interchange containing only a TA1 segment that indicates acceptance or error status for the whole Interchange received.
OnlyIfErrorTA1	If errors are found, generate a reply Interchange that contains only a TA1 segment that indicates error status for the whole Interchange received.
ISA14-TA1	If field ISA:14 of the incoming ISA header segment is set to 1, generate a reply Interchange containing only a TA1 segment; otherwise return nothing.
ISA14-OnlyIfErrorTA1	If errors are found and field ISA:14 of the incoming ISA header segment is set to 1, generate a reply Interchange containing only an error TA1 segment; otherwise return nothing.
Byte	Generate a reply consisting of a single character code: 'A' if the entire Interchange is accepted, 'R' if it is rejected due to one or more errors.

All of the options that relate to TA1 segments are used to force a TA1 segment to be generated, often as the only body segment of the reply interchange. This convention is used to represent the presence or absence of errors in the entire inbound Interchange. However, if an error is found in the incoming ISA or IEA that can only be reported in a TA1 segment, then a TA1 is generated even if the configured setting does not force a TA1 to appear.

Default Char Encoding

Specifies the character set of the input data. InterSystems IRIS automatically translates the characters from this character encoding. Supported values are `UTF-8` or any member of the `Latin`*n* family. The value `Native` means to use the native encoding of the InterSystems IRIS server.

Placing a @ (at sign) character at the beginning of this field means that the field identifies an internal NLS translation table instead of a logical character encoding.

The default depends on the adapter.

For information on character sets and translation tables, see Translation Tables.

Default Separators and Terminators

X12 documents typically come in batches consisting of an Interchange, which contains one or more Functional Groups, which each contain one or more Transaction Sets. The Interchange header (ISA) segment includes information about which separators and delimiters to use in parsing the document.

When an X12 Interchange is parsed, the separators are set to values found in the header, and the rest of the document is parsed using those values. When there is no Interchange header, the only obvious separator is the element separator (the character immediately following the segment name) and sometimes the segment terminator (if it is followed by **CRLF**). In this case, the system default separator values are used as the separators and the document is parsed based on those. The system default separators are:

- $C(30) for the repetition separator
- ":" for the component element separator
- and "~" for the segment terminator

It is possible to use the following settings, which appear in the **Additional Settings** area, to pass in different default values to use for these separators when there is no Interchange header:

- **Default Repetition Separator**
- **Default Element Component Separator**
- **Default Segment Terminator**

There are also parameters for these values in the EnsLib.EDI.X12.Document methods **ImportFromFile**, **ImportFromDevice**, **ImportFromLibraryStream**, **ImportFromString**, and **ImportFromIOStream**.

If a source document includes an invalid or duplicated separator, the system uses the corresponding separator in the **Additional Settings** area when it is defined or the corresponding system default separator.

Local Application ID

Colon-separated *LocalID*:*Qualifier* code that represents the facility and application that receive X12 documents via this business service. These are used to create reply document headers. The @ (at sign) character represents using the corresponding field from the incoming document. If your ID must contain a literal @ symbol, escape it with back slash: \@

The default value is:

```
EnsembleX12Service:03
```

Reply Mode

Specifies how to issue X12 reply documents (such as TA1 and 997). Options include:

- `Never` — Do not send back any reply.
- `Immediate` — Send a reply from the business service immediately upon receipt of an Interchange. This is the default.
- `Application` — Wait for a response from the target configuration item. When it arrives, relay the reply back to the sender. If validation fails or some other error occurs, generate an immediate reply according to the option selected for **BatchReplyType**.

Note: If an incoming ISA segment includes a field that too long or too short relative to the X12 standard, then that field is padded or truncated to meet the standard when it is copied to a reply document.

Reply Target Config Names

(**File** and **FTP** only) Specifies where the reply that the Business Service constructs, based on the validation that it performs, is to be sent. Usually the list contains one item, but it can be longer. The list can include business processes or business operations, or a combination of both.

Compare to Target Config Names.

SNIP Level

Relevant only when Validation Mode is SNIP.

- SNIP level 1 — segments are valid , segment order is valid, element attributes are valid, numeric data elements have numeric values, and message conforms to X12 rules.
- SNIP level 2 — meets HIPAA requirements, such as presence of required elements, non-use of elements marked as not used, and values conforming to the code tables.

Tolerate Newlines

True or False. If True, the business service processes an incoming X12 file without error, even if new lines have been inserted into the file after (or in place of) segment terminators to enhance readability. If False, these extra new lines trigger an error in parsing the file. The default is True.

Validation Mode

Two kinds of X12 validation are available:

- SNIP level validation — validates the X12 message according to the standards developed by the Workgroup for Electronic Data Exchange (WEDI) Strategic National Implementation Process (SNIP).
- Flag-based validation.

SNIP validation enables you to validate that:

- SNIP level 1 — segments are valid , segment order is valid, element attributes are valid, numeric data elements have numeric values, and message conforms to X12 rules.
- SNIP level 2 — meets HIPAA requirements, such as presence of required elements, non-use of elements marked as not used, and values conforming to the code tables.

Flag-based validation enables you to validate that:

- Required fields, including conditionally required fields, are present and that all fields are allowed by the schema.

- Conditionally excluded fields are not present.
- The number of fields within a segment and whether they are repeated as allowed by the schema.
- Data types for fields and components are correct.
- Field values conform to the code tables specified.
- Field and component values conform to length restrictions.

Validation

For details, see Validation in the following section.

Settings for X12 Business Processes

Provides reference information for settings of an X12 business process.

Summary

X12 business processes have the following settings:

Group	Settings	See
Basic Settings	Validation	*section in this topic*
	Business Rule Name	Settings for Routing Processes
Additional Settings	Alert On Bad Message, Bad Message Handler, Forward Generated Response To Target, Response Target Config Names, Response From, Response Timeout, Force Sync Send, Act On Transform Errors, Act On Validation Errors	Settings for Routing Processes
Development and Debugging	Rule Logging	Settings for Routing Processes

The remaining settings are common to all business processes. See Settings for All Business Processes.

Validation

The **Validation** setting of an X12 Document Router controls how the router validates the incoming message. If the incoming message fails the specified validation, InterSystems IRIS® reports the failure in the event log, and the X12 routing process passes the message to its bad message handler only; see the Bad Message Handler setting. If the message fails the specified validation but there is no bad message handler, an error is logged but the message is not sent to any target. If the message passes validation, the X12 routing process sends the message to the targets specified by the routing rules.

Ideally, you can use routing rules and data transformations to ensure each message is acceptable to the target system, and can, consequently, avoid using validation. This ensures that all messages are processed by the appropriate target. If you enable validation, InterSystems IRIS applies the validation tests before the routing rules. Any message that fails validation is not sent to a target based on the routing rules; it is sent only to the Bad Message Handler. However, there are some environments where X12 message validation is the preferred way to filter messages. For example, in the following situations, using X12 validation is a good choice:

- You are developing or debugging an interface and want to determine the kind of message variants that your system needs to handle.
- The target application cannot handle messages that have variances from the specification, and the routing rules and transformations cannot resolve those variances.
- There is a regulatory or other business requirement that the messages conform to the specification.

X12 validation does add overhead to the routing process. This overhead can be significant and can reduce the maximum load of messages that your production can handle.

The Validation property allows you to specify flags that control the following:

- Whether the message has a valid document type.
- Whether the message structure is validated.
- Whether the fields within the segments and the components within a composite structure conform to the schema.

If you specify an empty string as the Validation property value, the message router skips validation and routes all messages. When you create a new X12 routing process in the Management Portal, the **Validation** setting is initialized to an empty string.

Note: A message can pass validation and not conform exactly to the schema definition depending on the **Validation** flags specified.

The following table lists the X12 validation flags and describes how the routing process validates the message when each is specified.

Flag	Routing Process
d	Validation examines the DocType property of the document to see if it has a value.
m	Validation verifies that the document segment structure is well formed, and that it can be parsed using the schema identified in the DocType property of the document. This ensures that all required segments in the schema definition are included in the message and that the message does not contain any misplaced segments that are not allowed by the schema.
dm	Both d and m are active. This is the default for business processes (routers).
s	Validation enforces segment structure, which is the number and repetition of fields within a segment.
c	Validation enforces composite structures, which is the number of components.
x	Validation enforces relational conditions on segments and fields. Relational conditions indicate which fields or components are required or excluded based on the presence of other fields or components. At minimum, you must additionally specify s for segment-level validation. You can also specify c for component-level validation. **Note:** Relational conditions are present in SEF files and not in XSD files. For more information, see X12 Schema Distribution Files.
r	Validation enforces that the required elements are present. If s is also specified, then it tests if the required fields are present in the segment. If c is also specified, then it tests if the required components are specified in a composite structure. If neither s nor c is specified, then r has no effect.
u	Validation tests whether any elements that are marked as Not used are present. It tests for the presence of fields if s is also specified and tests for the presence of components if c is also specified. If any of these elements are present, validation fails. Note that this test cannot be performed if the only schema available is the new-style schema because it requires information stored only in the legacy schema.
l	Validation tests the length of elements. It tests the length of fields if s is also specified and tests the length of components if c is also specified.
t	Validation tests for the correct datatype of the elements. It tests the datatype of fields if s is also specified and tests the datatype of components if c is also specified.
v	Validation tests that the value of elements is allowed by the code table. It tests against the code table of fields if s is also specified and tests against the code table of components if c is also specified. Note that this test cannot be performed if the only schema available is the new-style schema because it requires information stored only in the legacy schema.
n	Equivalent to dmscrlt, which are all validation that can be performed relying only on the new-style schema.

Flag	Routing Process
a	Equivalent to `dmscrultv`, which are all validations.
e	Validation continues through the entire document even after encountering an error. If e is not specified, validation stops after encountering the first error.
(empty string)	Skips validation and routes all messages. This is the default for business services and operations.

Settings for X12 Business Operations

Provides reference information for settings of an X12 business operation.

Summary

X12 business operations have the following settings:

Group	Settings	See
Basic Settings	File Name	*section in this topic*
Additional Settings	Search Table Class, Default Char Encoding	Settings for Business Operations
	Auto Batch Parent Segs, Auto Batch Completion Timeout, No Fail While Disconnected, Separators, Validation, Reply Code Actions, Failure Timeout	*sections in this topic*

The remaining settings are either common to all business operations or are determined by the type of adapter. For information, see:

- Settings for All Business Operations
- Settings for the File Outbound Adapter
- Settings for the FTP Outbound Adapter
- Settings for the TCP Outbound Adapter

  EnsLib.X12.Adapter.TCPOutboundAdapter has the following settings configured appropriately for X12:

  - **Connect Timeout** has its usual default of 5 seconds, but has a maximum limit of 30,000 seconds.
  - **Get Reply** is set to False. This means the adapter waits to read a reply message back from the socket before returning.
  - **Response Timeout** has a default of 30 instead of its usual 15, and has a maximum limit of 30,000 seconds.
- Settings for the SOAP Outbound Adapter

Auto Batch Parent Segs

(**File** and **FTP** only) If True, when writing a document that has a batch parent, output the batch header segments first, then child documents, then follow up with the batch trailer segments when triggered by the final batch header document object or by a file name change. If False, omit headers and trailers and output child documents only. The default for X12 is True.

Auto Batch Completion Timeout

(**File** and **FTP** only) Providing a value for this setting enables InterSystems IRIS to add trailing segments to batch output which does not yet have the trailing segments necessary to make a complete batch document. The value specifies in seconds the period of time the document must remain unmodified before adding the trailing segments. This setting also ensures that trailing segments are always added between batches which are being written out to the same file. It impacts mainly productions in which the Service used **Batch Handling** = **Individual**, although routing a Transaction Set to a different business operation than its parent Group and Interchange would have prevented trailers from being added when **Batch Handling** = **Multi-Session Batch** or **Batch Handling** = **Single-Session Batch**. This setting has no impact on **Batch Handling** = **Whole Batch**.

0 means never timeout. This setting is only relevant if **Auto Batch Parent Segs** is True.

Default Char Encoding

Specifies the desired character set of output data. InterSystems IRIS® automatically translates the characters to this character encoding. For X12 output, the default is `Latin1`. See Default Char Encoding in Settings for X12 Business Services.

Failure Timeout

The number of seconds during which to continue retry attempts. After this number of seconds has elapsed, the business operation gives up and returns an error code. X12 business operations automatically set this value to –1 for *never time out* to ensure that no X12 document is skipped.

File Name

(**File** and **FTP** only) Output file name. This setting can include time stamp specifiers. If you leave **File Name** blank, the default value is `%f_%Q` where:

- `%f` is the name of the data source, in this case the input filename
- _ is the literal underscore character, which appears in the output filename
- `%Q` indicates ODBC format date and time

In substituting a value for the format code `%f`, InterSystems IRIS strips out any of the characters |,?,\,/,:,[,],<,>,&,,,,;,NUL,BEL,TAB,CR,LF, replacing spaces with underscores (_), slashes (/) with hyphens (-), and colons (:) with dots (.).

For full details about time stamp conventions, including a variety of codes you can use instead of the default `%f_%Q`, see Time Stamp Specifications for Filenames.

No Fail While Disconnected

(**TCP** only) If True, suspend counting seconds toward the **Failure Timeout** while disconnected from the TCP server. This setting does not apply if **Failure Timeout** is –1 or if **Stay Connected** is 0.

Reply Code Actions

(**TCP** only) When the adapter setting **Get Reply** is True, this setting allows you to supply a comma-separated list of code-action pairs, specifying which action the business operation takes on receipt of various types of acknowledgment documents. The format of the list is:

code`=`*action*`,`*code*`=`*action*`, ...` *code*`=`*action*

Where *code* represents a literal value found in field TA1:4, AK5:1, or AK9:1 of the acknowledgment document. The following table lists the expected values for *code*.

Code	Meaning
`A`	Accepted
`E`	Accepted, But Errors Were Noted
`R`	Rejected
`M`	Rejected; Message Authentication Code (MAC) Failed
`W`	Rejected; Failed Validity Tests
`X`	Rejected; Content Decryption Failed
`~`	The tilde character matches replies that do not contain a TA1, AK5 or AK9 segment

Code	Meaning
_	The underscore character matches replies with an empty value in the field. An empty or whitespace code value is the same as _
*	The asterisk character matches any value not matched otherwise (default=S)
I?	Matches the case in which the reply ControlId does not match the ControlId of the original document

The following values for *action* may be used alone or combined to form strings. S is the default *action* if no other is given, except for A whose default action is C:

Action	Meaning
C	Treat the document as Completed OK. Code A has a default action of C.
W	Log a warning but treat the document as Completed OK.
R	Retry the document according to the configured RetryInterval and FailureTimeout; finally Fail unless a different action is also specified
S	Suspend the document, log an error, and move on to try the next document. S is the default action for all codes except for code A, which has a default action of C.
D	Disable the business operation, log an error and restore the outbound document to the front of the business operation's queue
F	Fail with an error and move on to try the next document

The default value for this setting string is:

```
A=C,*=S,~=S,I?=W
```

This means:

- `A=C` — When the action is accepted, treat the document as Completed OK.
- `I?=W` — When the reply ControlId does not match the ControlId of the original document, log a warning but treat the document as Completed OK.
- `*=S,~=S` — In all other cases, including when replies that do not contain a TA1, AK5 or AK9 segment, suspend the document, log an error, and move on to try the next document.

Separators

A string of separator characters which InterSystems IRIS assigns to X12 separators in left to right order as described below.

An X12 document uses special characters to organize its raw contents. These characters may vary from one clinical application to another. For non-empty values of **Separators**, positions 1 through 3 (left to right) are interpreted as follows:

1. Data Element Separator (ES)
2. Component Separator (CS)
3. Data Element Repeat Separator (RS)

The default values for positions 1 through 3 are:

1. `*` (asterisk)
2. `:` (colon)
3. `^` (caret) for SOAP-based operations and `\a` (record separator) for all other operations

For **Separators**, you must supply a string of three characters which InterSystems IRIS assigns to X12 separators in left to right order: ES, CS, RS, as described in the previous list.

Any characters in positions 4 through 6 override the default segment terminator character, which is ~ (tilde). You may specify from 0 to 3 characters in positions 4 through 6 using the following:

- `\r` for the carriage return (ASCII 13)
- `\n` for the line feed (ASCII 10)
- `\a` for the array record separator (ASCII 30)

You can use `\x` in positions 1 through 3 if you need to specify segment terminators in positions 4 and higher but want your output documents to use fewer than 3 separators. Separators designated by `\x` in positions 1 through 3 are not used. The purpose of `\x` is simply to extend the length of the list of separators so that position 4 is interpreted correctly as the first segment terminator.

If the **Separators** string is empty, the default is to use the current default separators and segment terminators for X12, plus a carriage return (ASCII 13) and line feed (ASCII 10), for example:

```
*:\a~\r\n
```

Validation

Any non-empty string triggers basic validation of the outgoing document. If the **Validation** field is left empty, no validation of the outgoing document is performed.