Etasoft
Building blocks of integration

EDI Validator User's Manual for version 2.x

Main website http://www.etasoft.com
EDI Validator website http://www.xtranslator.com

Purpose

EDI Validator is data validation package for EDI X12 format messages. While it is completely stand alone and independent, it is recommended to be used with translator package in order to validate incoming and outgoing data.

Additional Documentation

User Manual describes basic use of the EDI Validation Editor and other validation utilities. You can find additional documentation and sample source code for Developer SDK in software installation directory.

Requirements

Table lists minimum hardware and software requirements for the application to run:

CPU	Pentium 700 MHz
RAM	256Mbt
HDD	1.2Gbt
Operating System	Windows 2000/XP/Vista/7, Windows Server 2003

If you are working with big input files recommended configuration is:

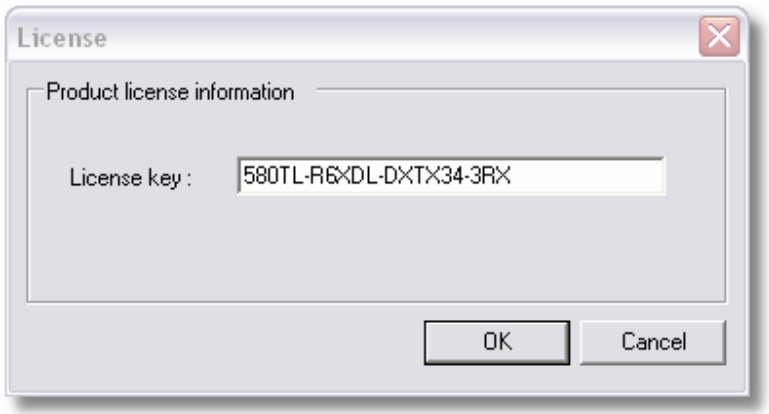
CPU	Pentium 2GHz
RAM	2Gbt
HDD	1.2Gbt
Operating System	Windows 2000/XP/Vista/7, Windows Server 2003

License

You can distribute evaluation version free of charge.

Licensing is based on number of installations. Retail version with setup license key can be installed on as many machines as many licenses have been purchased. You can also buy unlimited site license and install the software on unlimited number of computers in organization. Visit our website to find more licensing details.

You can register and buy licenses online. After the purchase email will be send to you with the license key.



Enter your license key in Product License screen.

Terminology

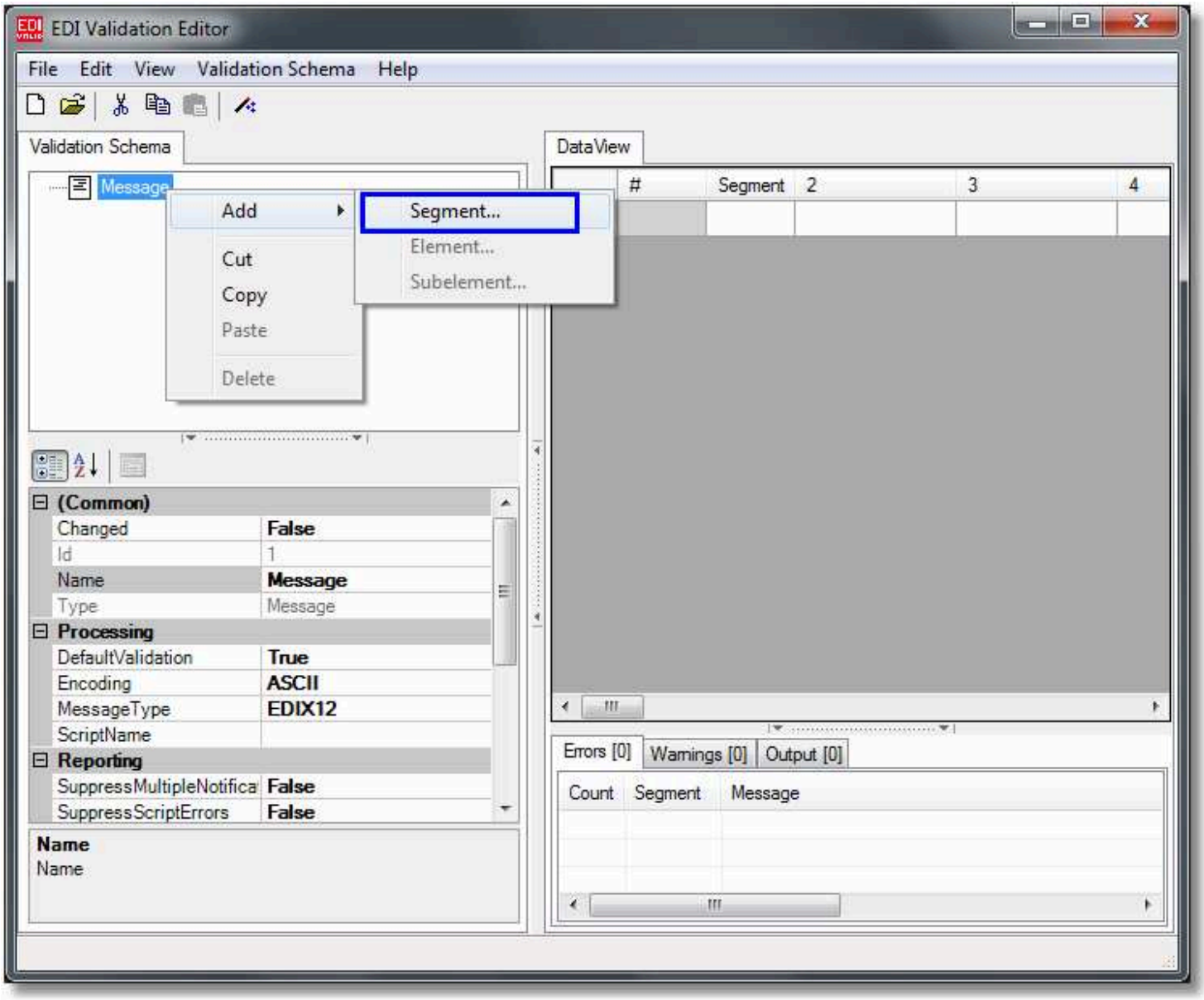
List of definitions and they explanations:

Definition	Message Type	Meaning
Message	All types	File or data stream to be processed.

Segment	EDI X12	EDI segment, such as ISA, GE, ST.
Element	EDI X12	EDI element, block of data inside of segment, terminated by EDI separator
Subelement	EDI X12	EDI component sub element, block of data inside of element

Validation Schema

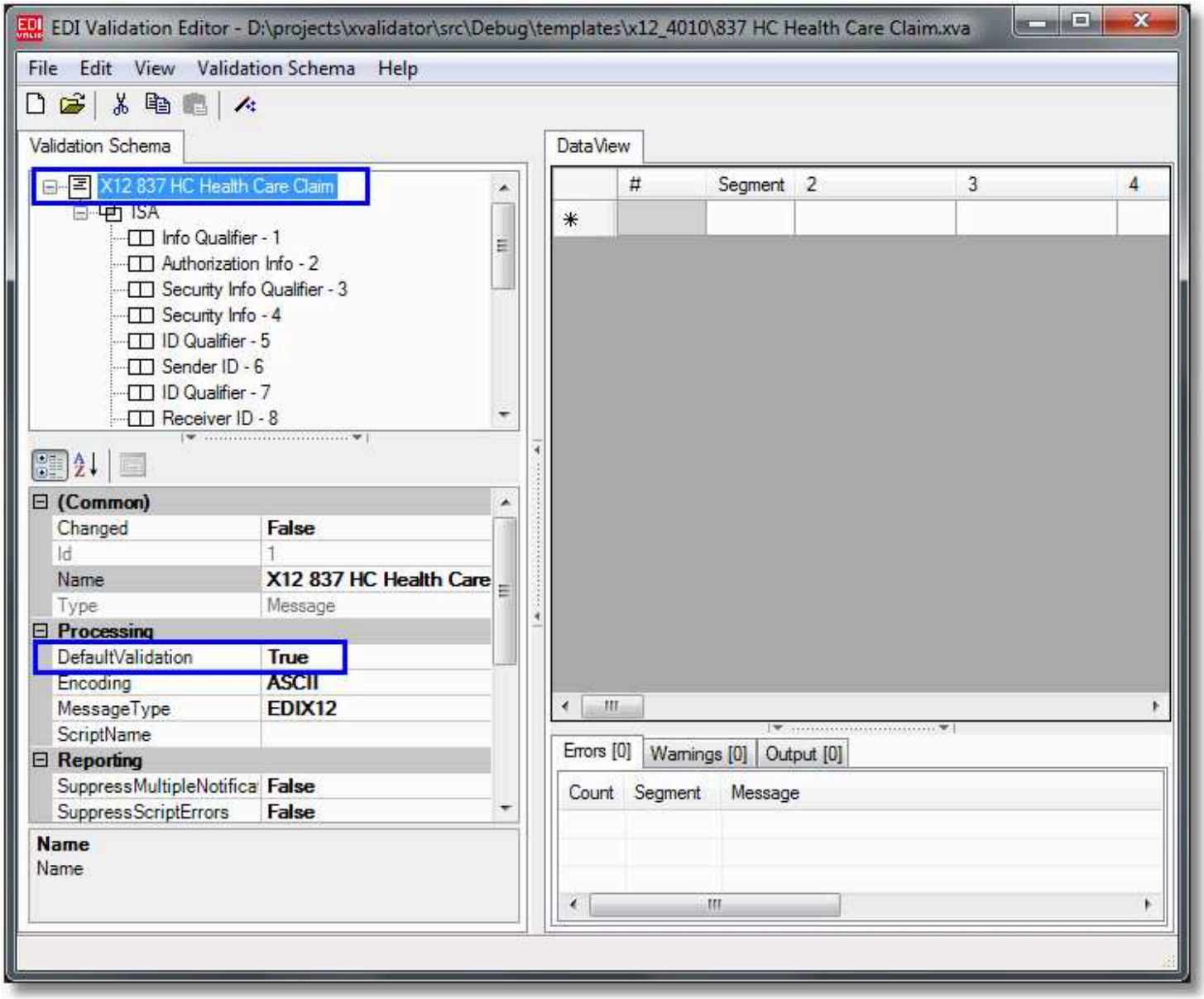
EDI X12 message has to be defined for validation to work. Message should contain all expected segments, elements and sub elements. If some segments are missing, they will generate warning or error messages. Segments can be added manually using Add->Segment menu option or using Templates.



Adding segment manually.

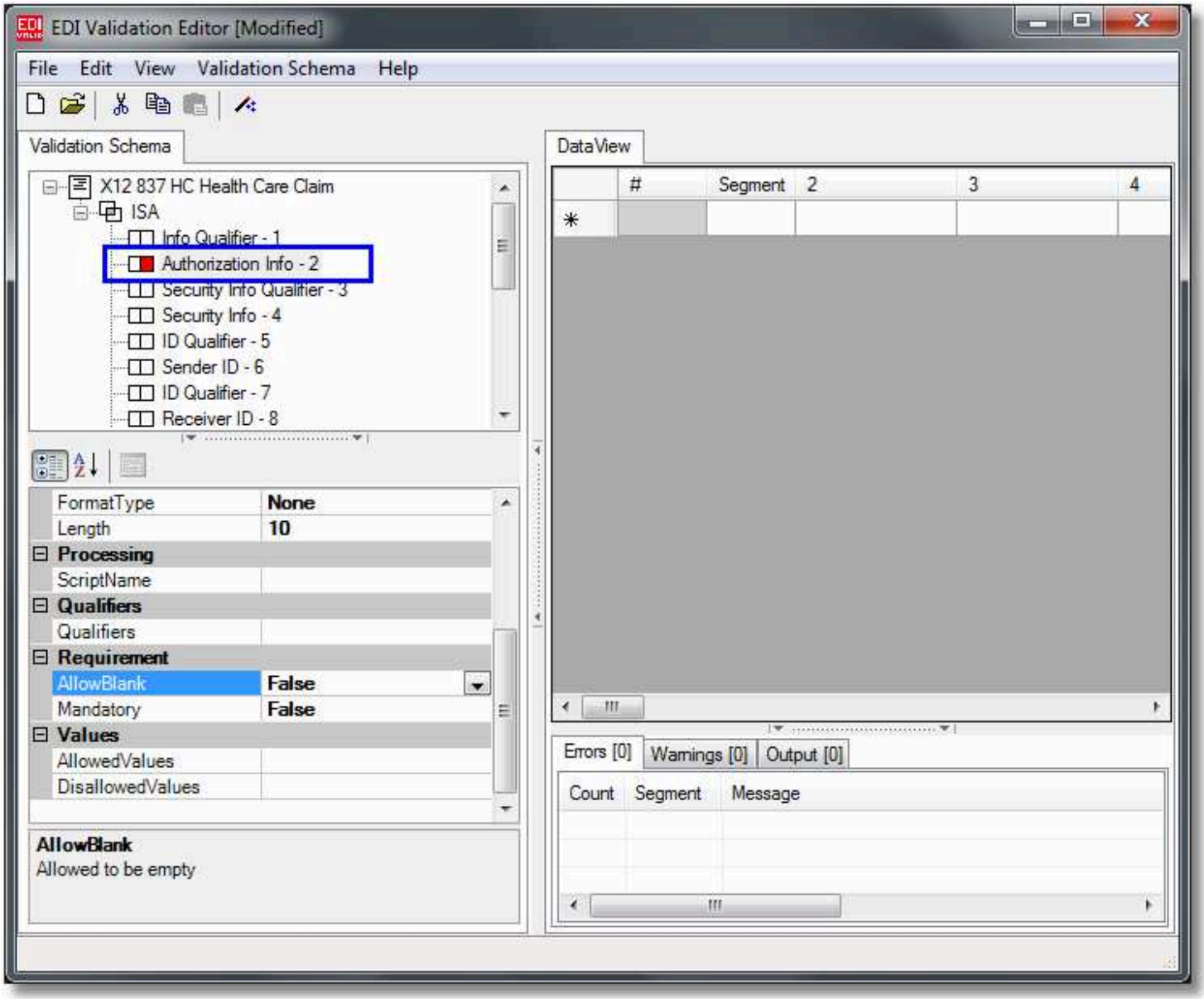
Segments should be added manually only if there is no pre-built template already supplied with the product. All of the templates are accessible via Project->Templates menu. When adding segments manually Tag property has to be supplied in order for

validator to find and identify segment in input message.



DefaultValidation properties.

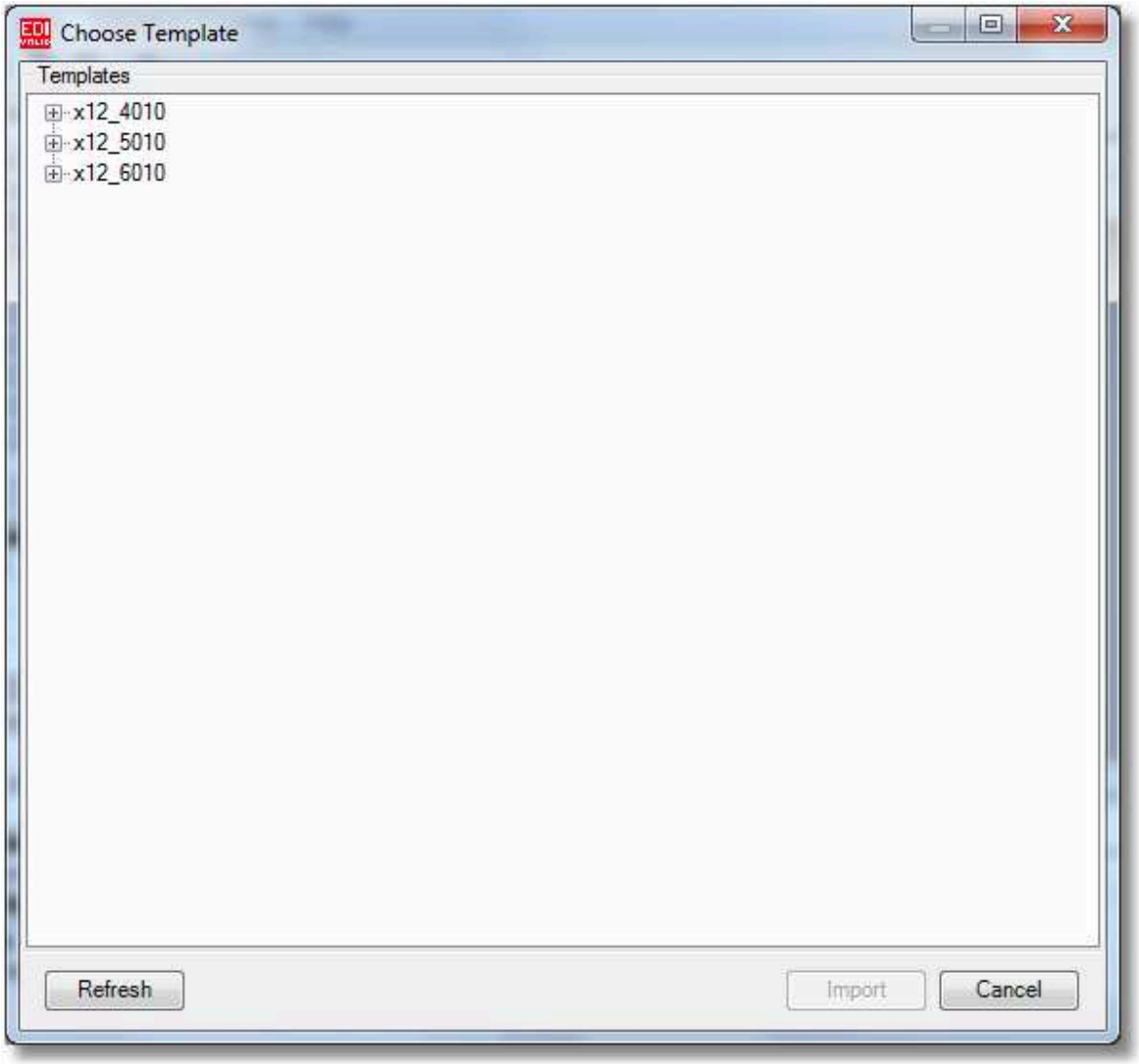
When set to true, DefaultValidation property turns on basic validation for EDI X12 messages. If your EDI X12 message does not have standard enveloping segments (like ISA, GS, GE, IEA, etc.) then this option should be turned off, as those segments are primary basic validation targets. Please check “Default Validation” chapter for more details.



Whenever any processing property changes Changed property is set to True and object icon is marked red.

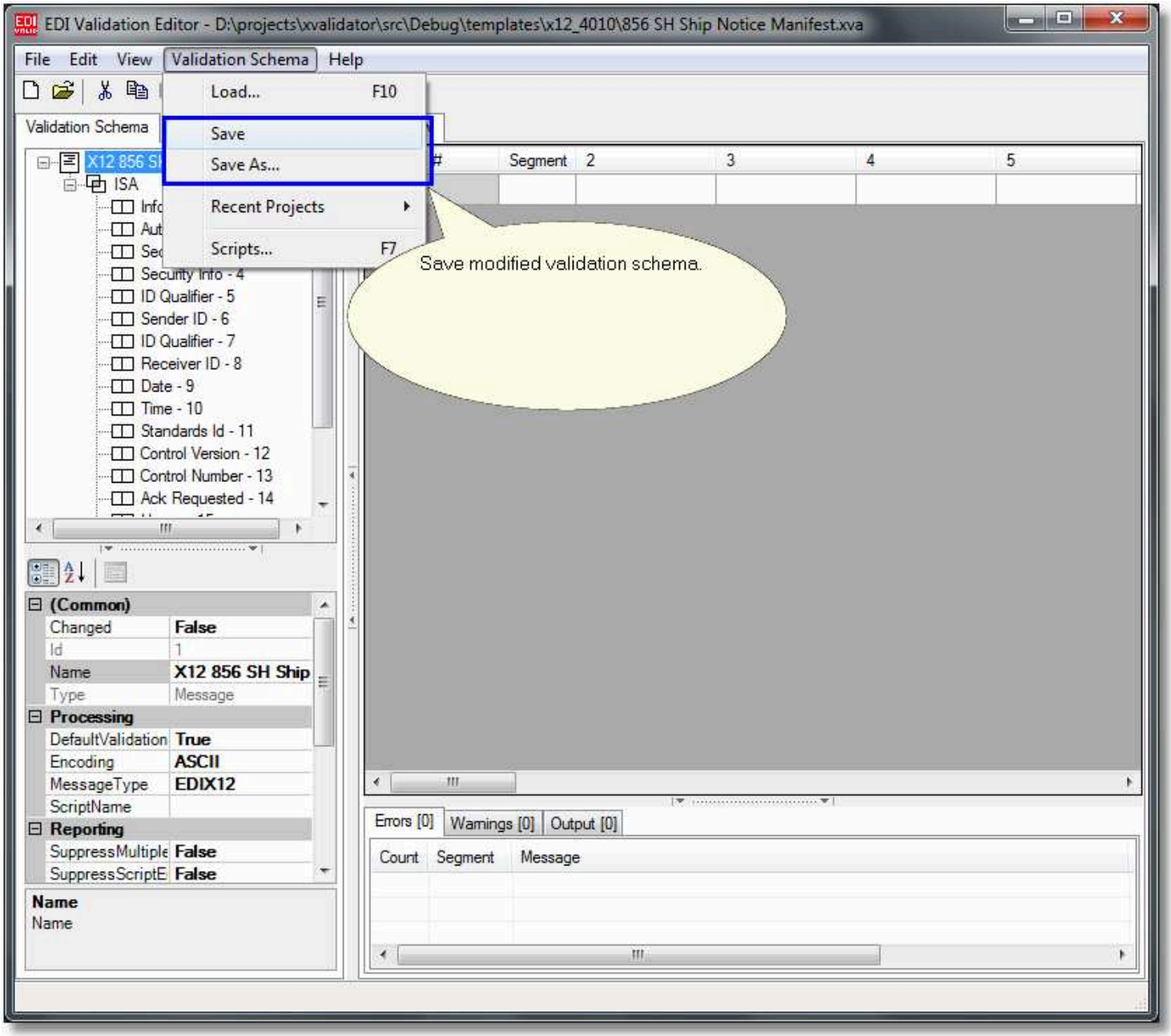
Validation Schema Templates

Most common EDI X12 standard messages are included in the package. You can load template using “Templates…” menu or by pressing F10 on your keyboard. Template directory is scanned and all template names appear in the list.



Standard EDI X12 templates included in the package.

Pre-built templates come without sub elements defined. Sub elements have to be added manually in order for the validator to be able to validate them. If sub element is received but not defined in the schema, validator will produce warning message about it.



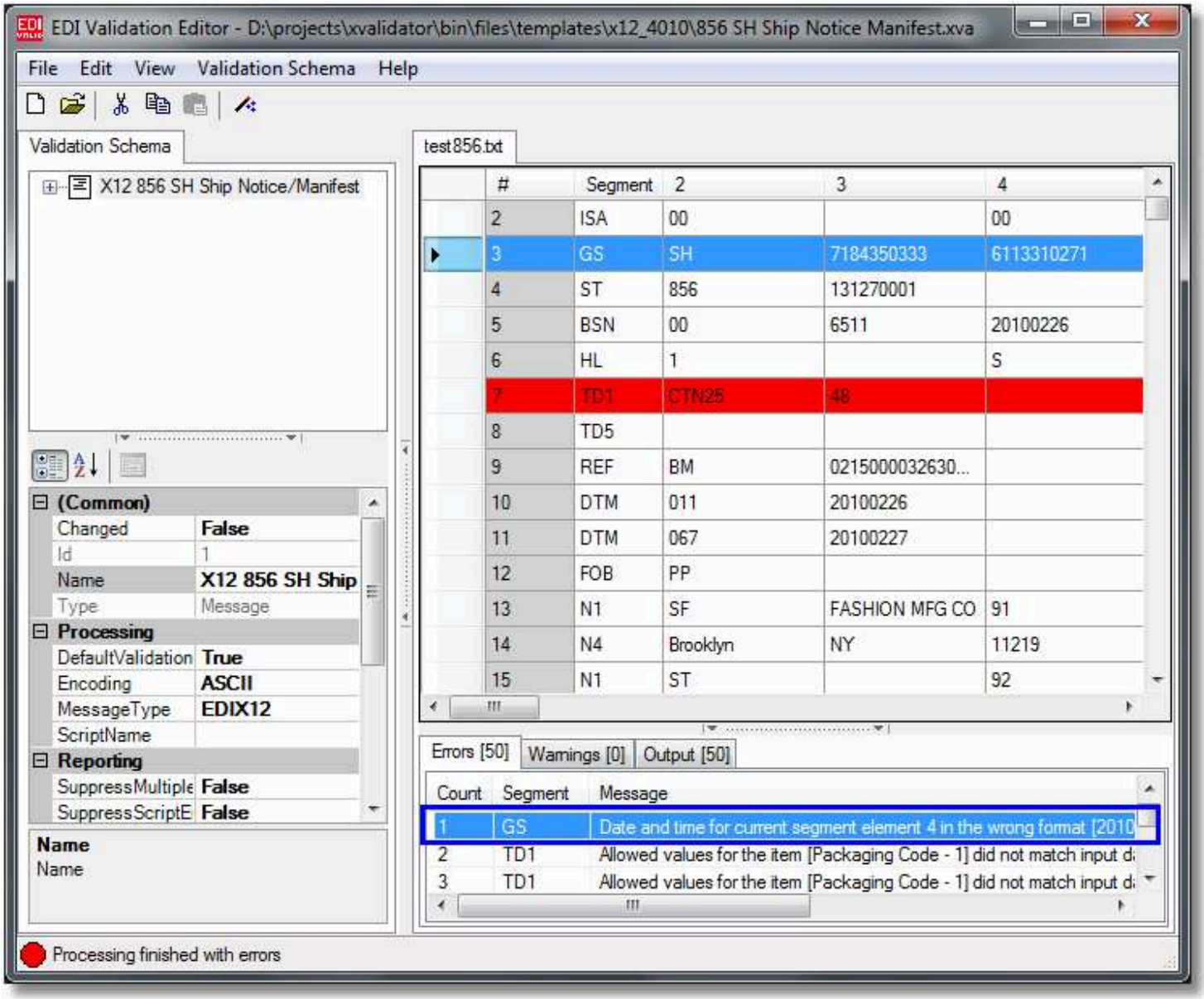
Save modified template to directory of your choice. This allows you to preserve original templates, and changes you make are specific to your business rules.

Processing

Internally processing is divided in two stages:

1. Low-level parser reads all the incoming data and parses it into segments, elements and sub elements.
2. Validator uses definitions, objects and properties set by Validation Editor and performs validation.

All the errors and warnings are displayed in the lists at the bottom of the screen.

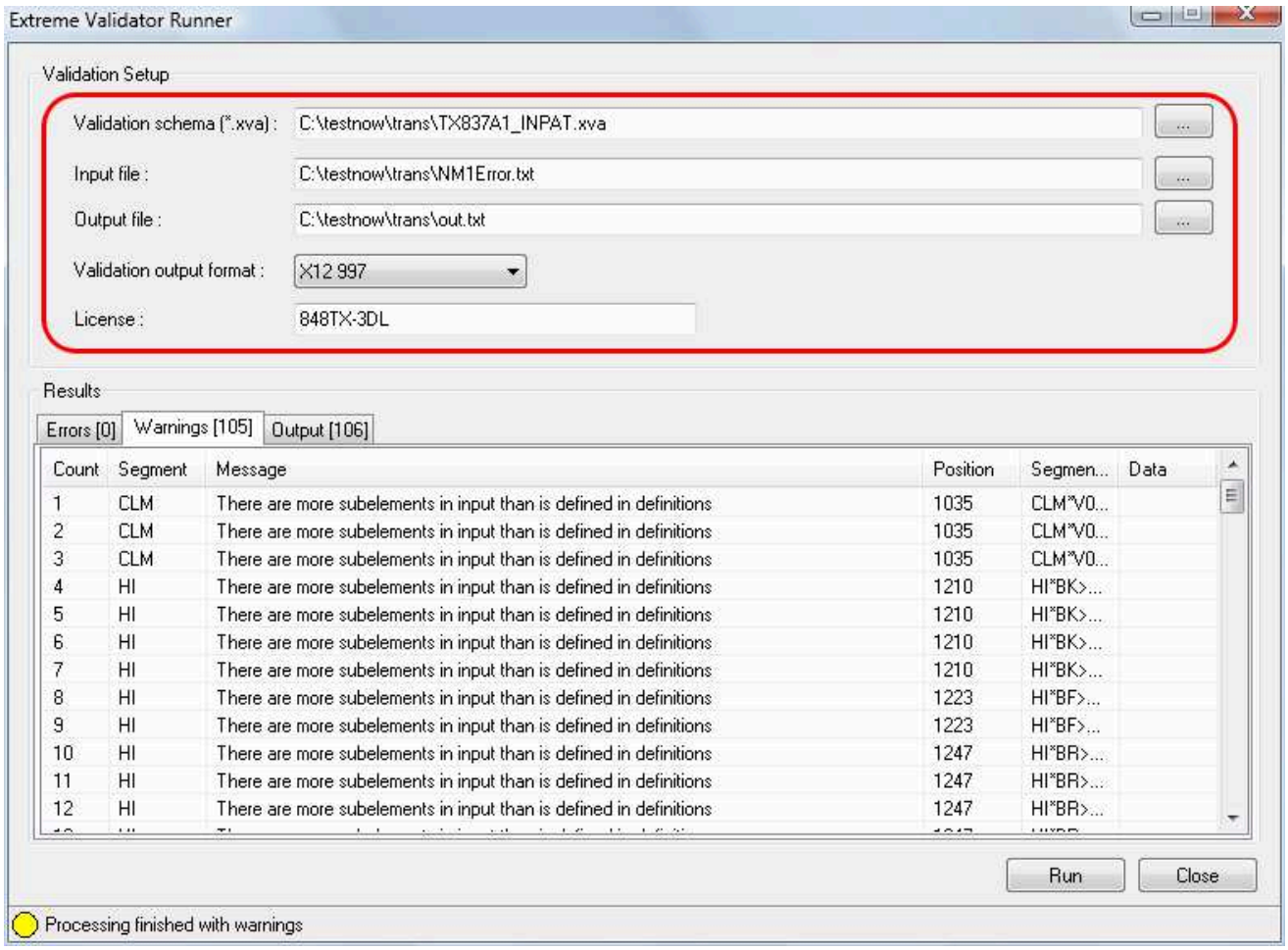


Typical processing screen when validation fails with errors. Error lines indicated in red and warning lines indicated in yellow.

You can click on error in the bottom list and data view will jump to the EDI X12 segment line with error.

You can minimize number of errors and warnings reported by setting properties SuppressMultipleNotifications and SuppressScriptErrors to True. When SuppressMultipleNotifications is set to True, validator will report only one error per segment. Let say you have number of elements wrong in ST segment, but only first error detected will be displayed. SuppressScriptErrors will not show errors produced by custom script executions, even if script fails with runtime error or fails to load.

You can use additional program called “validrun” to execute validations without use of EDI Validation Editor.



There is Validation Runner “validrun.exe” in action.

Batch Processing

Validation can be executed as time scheduled job, run as part of other batch process or be executed as external job to do the validation. This makes it very flexible. Validator command line utility can produce output in plain text or XML format. First two parameters are required and they must be included on the command line. Parameters are listed with they priority and expected order:

1. Mandatory. Path to definitions file with extension *.xva.
2. Mandatory. License key in form License=my_license_key
3. Mandatory. Input file in form Input=C:\test\testfile.txt. You can use wildcard to process multiple files like this: Input=C:\test\test*.txt, and it will pick all text files starting with word “test”.
4. Optional. OutputFormat=Plain or OutputFormat=XML
5. Optional. Output=C:\some_directory\errors_warnings.txt
6. Optional. OnError=Stop

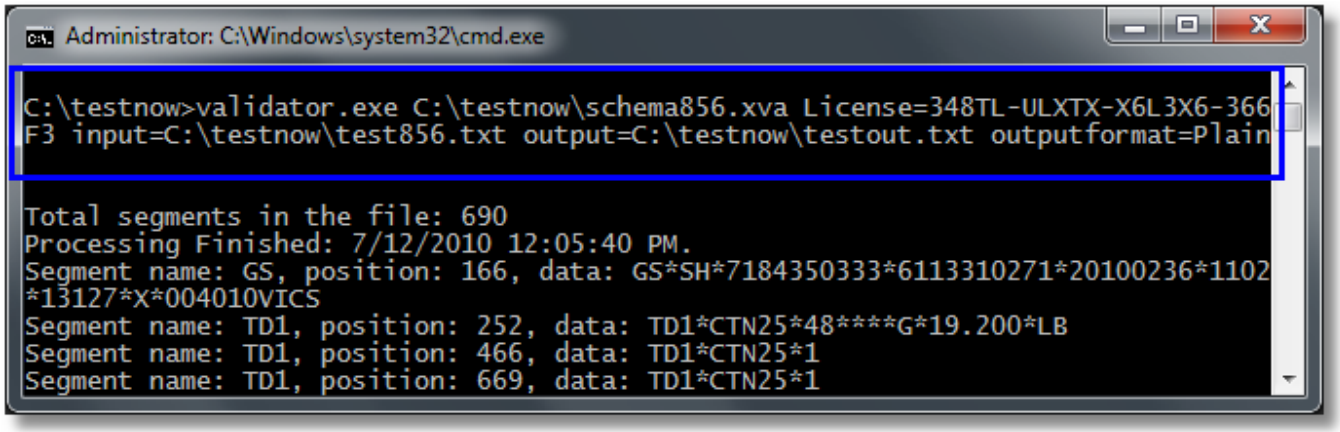
Output file name may contain following macros:
%Count% - this macro will increment for each input file if multiple input files are processed (when * wildcard is used in the input file).
%SystemDate% - this macro will produce current system date in the file name.
%SystemDateTime% - this macro will produce current system date and time in the file name.

Example: if parameter Output is set to C:\some_directory\output%SystemDate%.txt then produced file name will be C:\some_directory\output20061204.txt.

You can use unexpired license key from Definition Editor Help->Product License screen. Also use OutputFormat to output errors and warnings in plain text or XML formats and output them on the screen (default) or write them into the file specified with Output switch in command line.

Recommendation: try to use directories and file names without space characters in them. Spaces provide additional challenges, and when entered as command line parameters they have to be enclosed in quotes. If quotes are omitted or missed spaces will split command line parameters and command will fail to run.

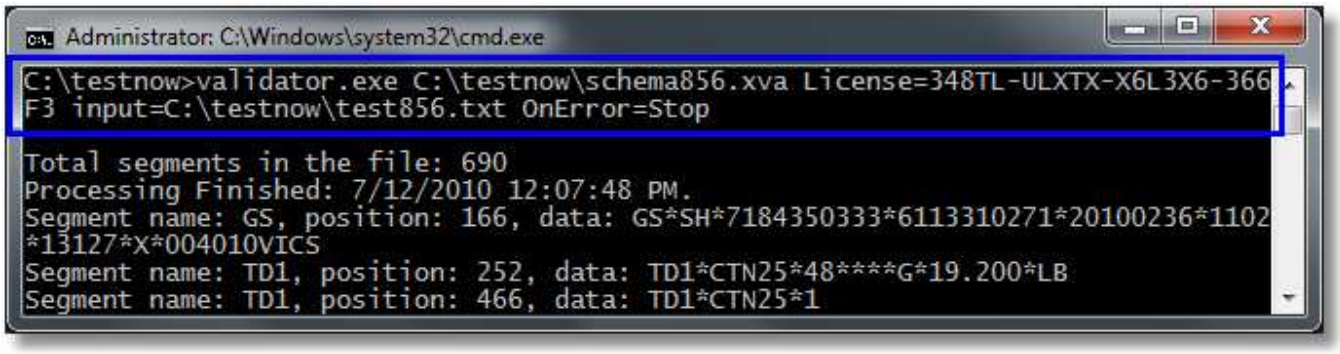
Example 1:
validator.exe C:\testnow\schema856.xva License=348TL-ULXTX-X6L3X6-366F3 input=C:\testnow\test856.txt output=C:\testnow\testout.txt outputformat=Plain



Batch processing screen with Output and OutputFormat specified.

Example 2:
validator.exe C:\testnow\schema856.xva License=348TL-ULXTX-X6L3X6-366F3 input=C:\testnow\test856.txt OnError=Stop

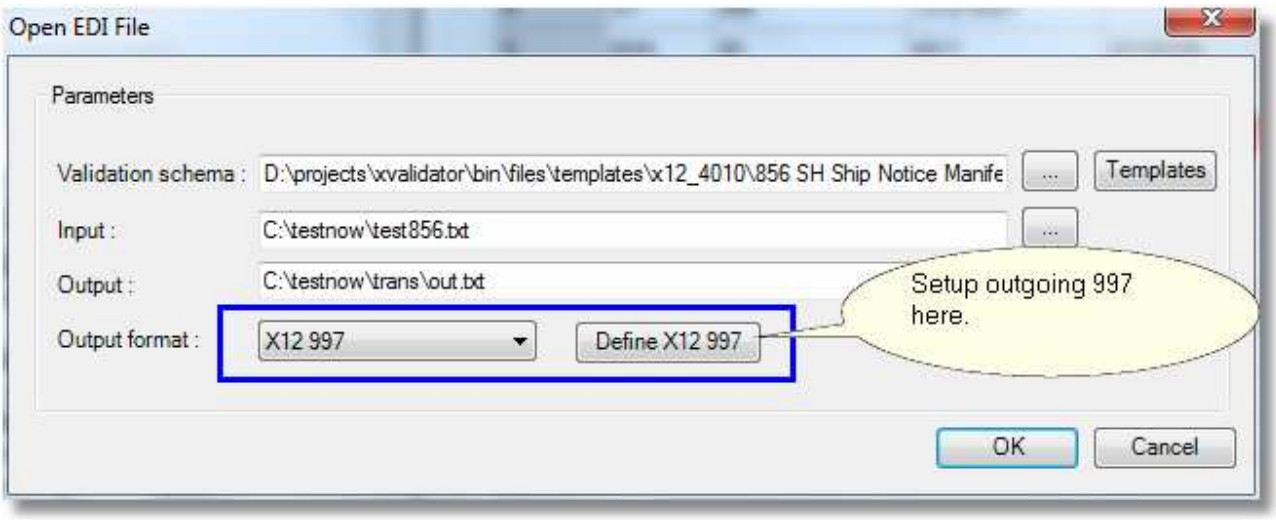
In Example 2 Output and OutputFormat is omitted. But OnError flag is added. This way output is provided only to the screen but if validation error happens, program stops execution and waits for user input.



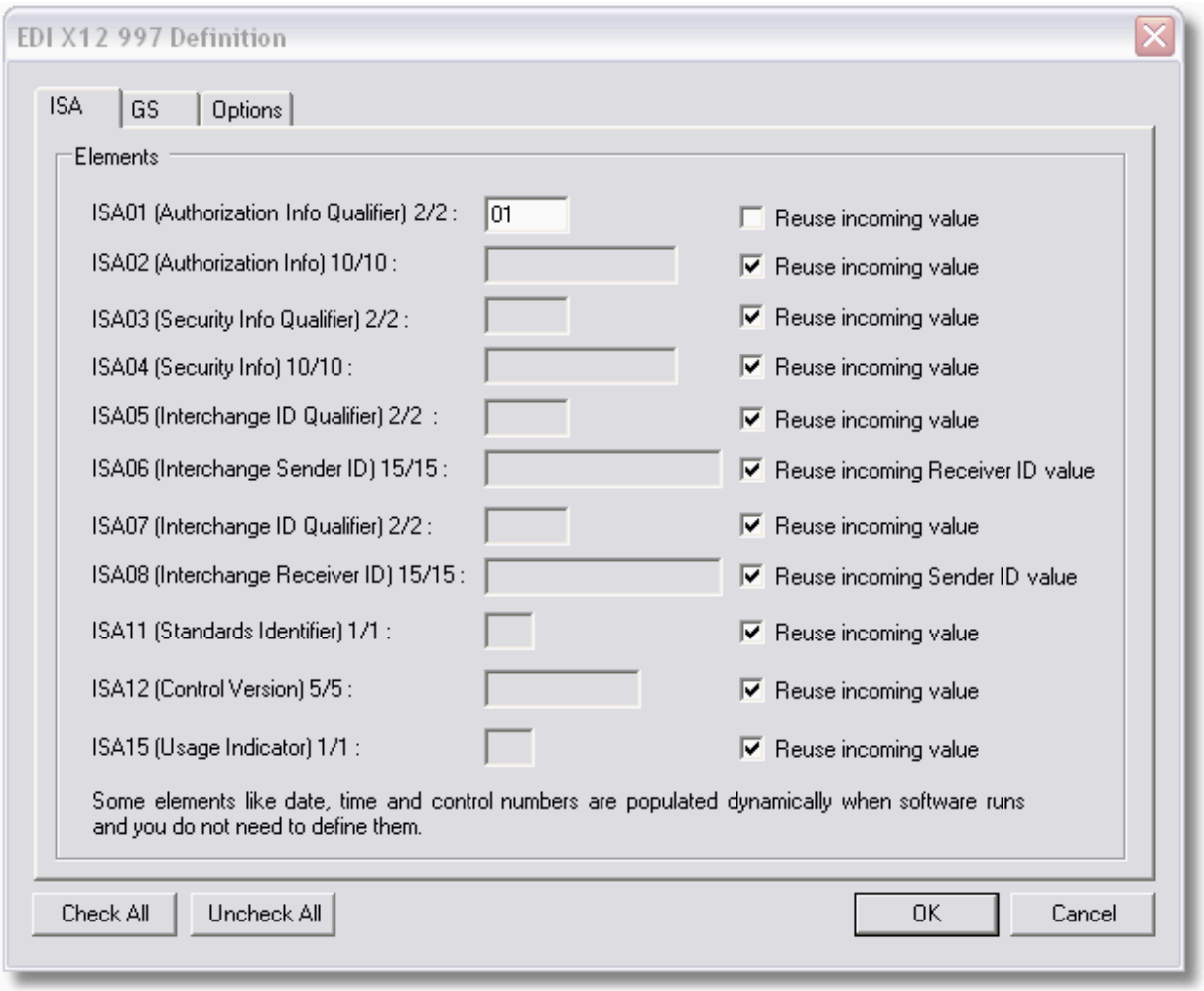
Batch processing screen with OnError flag but without Output and OutputFormat flags specified.

Output X12 997 or 999 acknowledgement

Validator can produce X12 997 or 999 output as a result of validation of EDI X12 files. You can define X12 997 and 999 output by using setup window under “Run With Parameters..” screen.



Define outgoing X12 997 via this screen.



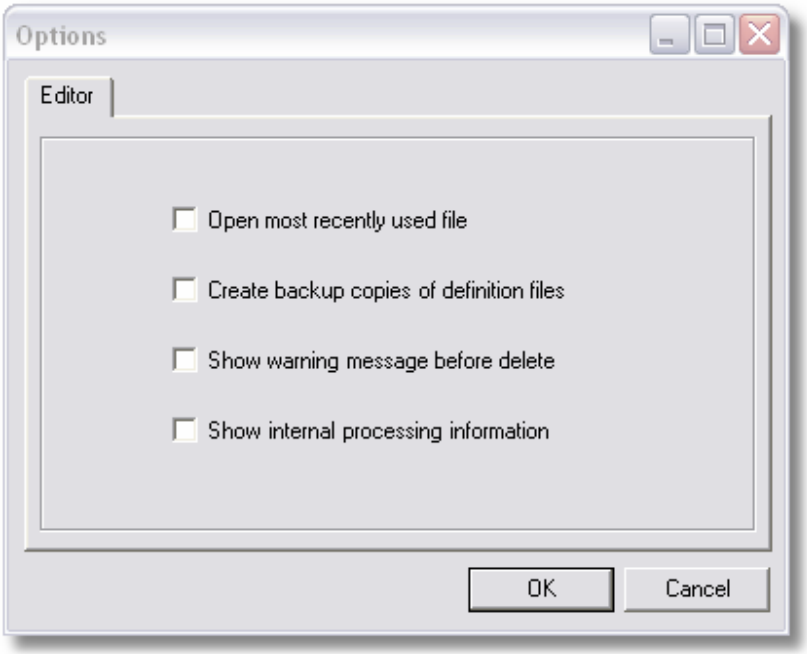
X12 997 and 999 processing is setup to reuse incoming values by default. But you can overwrite this behavior by unchecking “Reuse..” checkbox and providing your own constant value. X12 997 and 999 default processing will also swap Receiver and Sender Ids from incoming file and reuse separators (delimiters) from the incoming file.
Control number for 997 ISA, GS, ST, SE, GE and IEA segments will be generated and incremented each time validation runs. In order to reset generated values, delete file “997controls.txt” and “999controls.txt” from software installation directory.

Once you define X12 997 and 999 processing, save definitions. Now you can also generate 997 or 999 via command line. Simply use OutputFormat=997 or OutputFormat=999 as a parameter:

validator.exe c:\testnow\xvalid\test837P1.xva Input=c:\testnow\xvalid\837P1.txt License=848TL-RUFF4-TUUDDR-6RLFF Output=c:\testnow\xvalid\837P1_997.txt OutputFormat=997

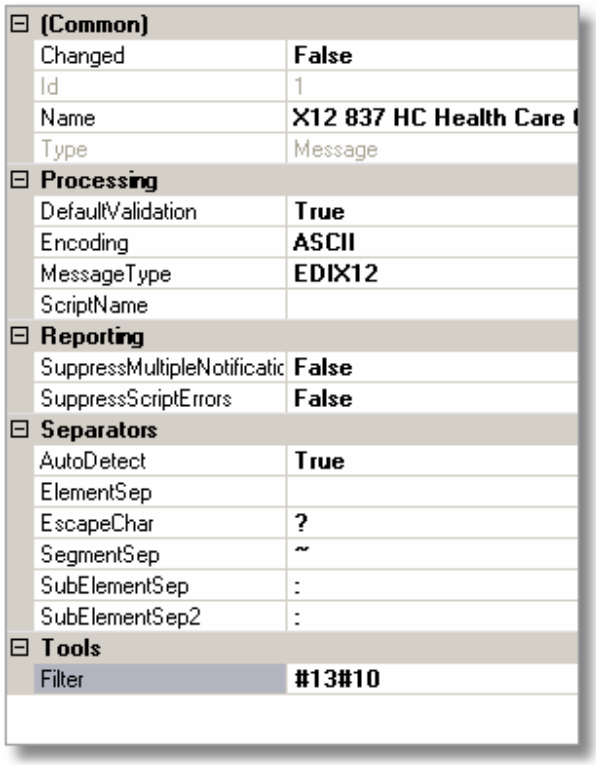
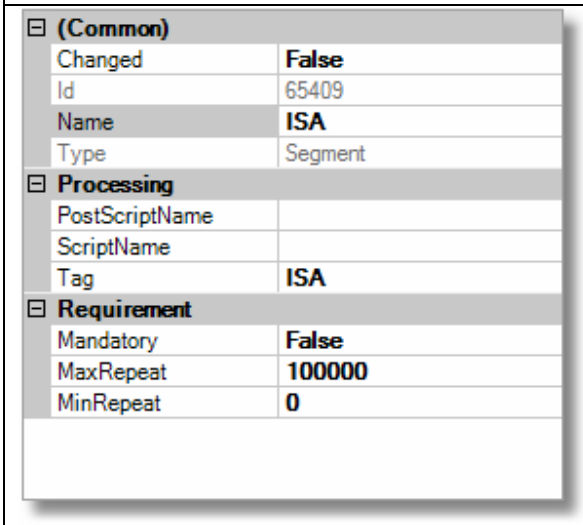
Options

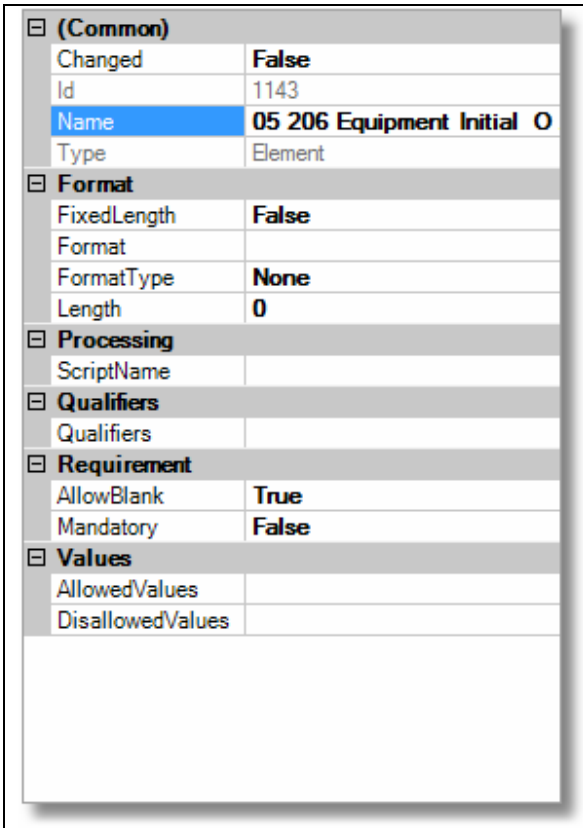
Open most recently used file option will open most recently loaded or saved file.
Create backup copies of definition files will create extra backup copy of current file.
Show warnings message before delete will show warning message when definition item is about to be deleted.
Show internal processing information will show internal validation processing information messages in Output pane. This option can be helpful when working with very complex validations.



Use Options screen to set Schema Editor options.

Properties

Screen shot of properties	*Properties*
 Message properties	**Changed** property is used to indicate items with changed property values in the definitions tree. **Id** is property used internally by validator. **Name** is object name displayed on the screen. **Type** is object type. Can be Message, Segment, Element or Subelement. **DefaultValidation** is a flag to use build in basic validation for EDI X12 or EDIFACT (see chapter Default Validation) **Encoding** defaults to ASCII but can be changed to European_8bit if is used to validate inputs with accented European characters. **MessageType** can be EDIX12 or EDIFACT. **ScriptName** script to be executed when message processing starts. **SuppressMultipleNotifications** will suppress multiple error or warning messages about the same segment. **SuppressScriptErrors** will suppress errors raised during script execution. **AutoDetect** will attempt to learn separators from incoming file. AutoDetect may fail if EDI X12 message does not have ISA segment or EDIFACT does not have UNB segment at the top of the message. **ElementSep, EscapeChar, SegmentSep, SubElementSep, SubElementSep2** various separators. They have to be set if AutoDetect is set to false. **Filter** can be used to filter garbage from input files. For example: to filter unwanted carriage return and line feed characters enter #13#10 in Filter property. Important: character filtering may shift segment positions reported in errors and warnings.
 Segment properties	Common properties explained in Message section above. **ScriptName** script to be executed when segment processing starts. **PostScriptName** script to be executed when segment processing ends. **Tag** is specific character string that identifies segment. **Mandatory** should be set to true if segment is mandatory. **MaxRepeat** maximum number of occurences for this segment. **MinRepeat** minimum number of occurences for this segment.
	Common properties explained in Message section above. **FixedLength** element has fixed length and uses Length property to validate it. **Length** used with FixedLength to validate items length. **FormatType** can be used to validate input data against various formats defined in Format property. **Format** specific format to validate input data. **ScriptName** script to be executed when element processing starts.



Element and subelement properties

Qualifiers if any qualifier is entered in this property validation will match not just segment name (Tag) but also value of the element that has qualifier set.
AllowBlank by default will allow incoming data to have no value.
Mandatory if set to true, will have to be always present.
AllowedValues list of allowed values separated by comma.
DisallowedValues list of disallowed values separated by comma.

FormatType and Format Properties

FormatType works with Format property but in some cases Format can be left blank.

FormatType	Format	Description
Integer	N/A (not applicable)	Validates if input data is an integer number
Numeric	N/A	Validates if input data is a number
RegularExpression	Regular expression	Validates if input data matches defined regular expression
DateTime	(See next chapter on FormatType - DateTime)	

FormatType DateTime

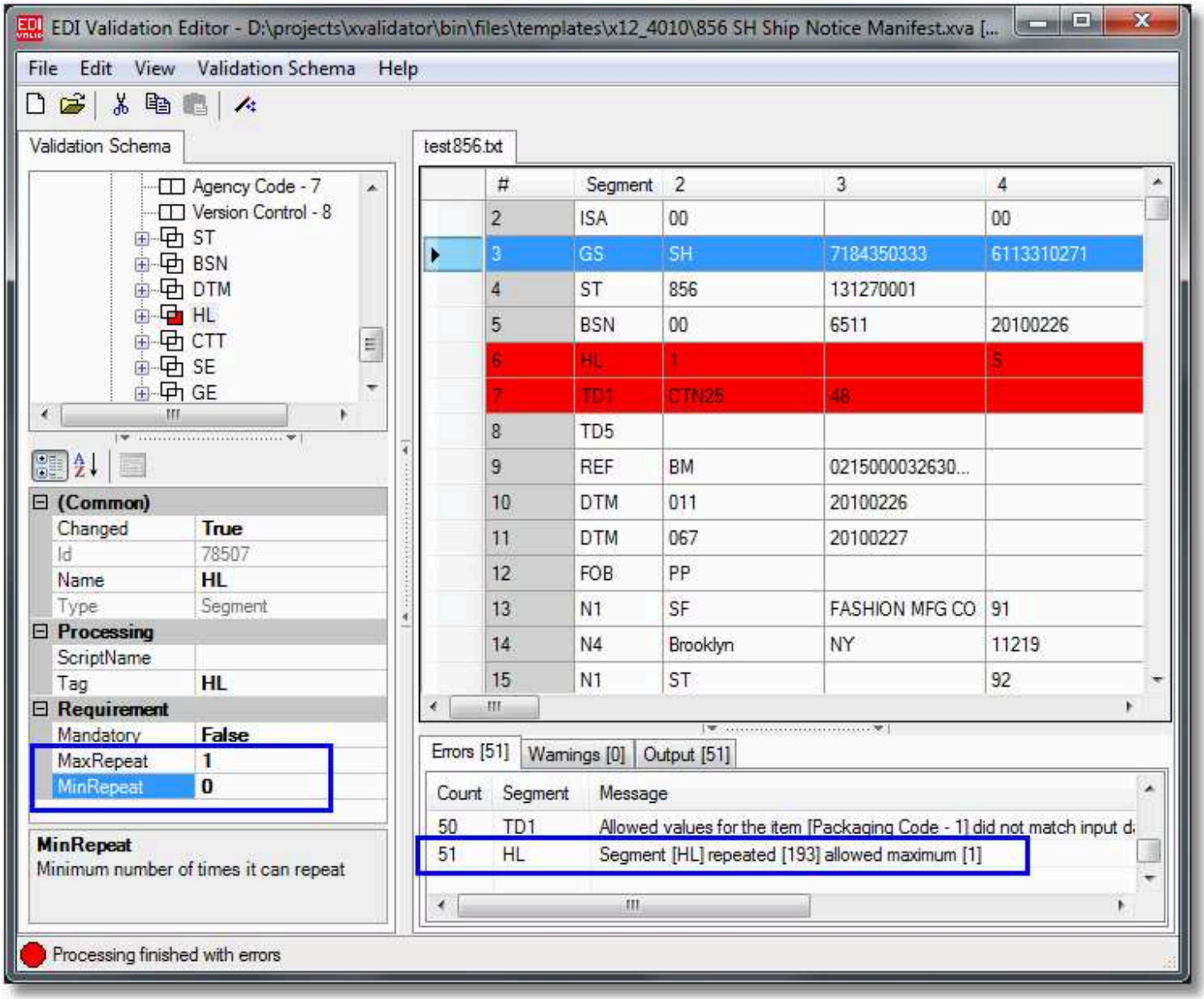
When FormatType is set to DateTime, Format property may contain combination of specific characters to validate dates and times.
Example: 12-JAN-2002 can be validated with Format set to d-MMM-yyyy
Example2: 12-2-04 can be validated with Format set to d-M-yy
Example3: 02-04-05 1314 can be validated with Format set to dd-MM-yy HHmm.

d The day of the month. Single-digit days will not have a leading zero.
dd The day of the month. Single-digit days will have a leading zero.
ddd The abbreviated name of the day of the week, as defined in AbbreviatedDayNames.
dddd The full name of the day of the week, as defined in DayNames.
M The numeric month. Single-digit months will not have a leading zero.
MM The numeric month. Single-digit months will have a leading zero.
MMM The abbreviated name of the month, as defined in AbbreviatedMonthNames.
MMMM The full name of the month, as defined in MonthNames.
y The year without the century. If the year without the century is less than 10, the year is displayed with no leading zero.

yy The year without the century. If the year without the century is less than 10, the year is displayed with a leading zero.
yyyy The year in four digits, including the century.
gg The period or era. This pattern is ignored if the date to be formatted does not have an associated period or era string.
h The hour in a 12-hour clock. Single-digit hours will not have a leading zero.
hh The hour in a 12-hour clock. Single-digit hours will have a leading zero.
H The hour in a 24-hour clock. Single-digit hours will not have a leading zero.
HH The hour in a 24-hour clock. Single-digit hours will have a leading zero.
m The minute. Single-digit minutes will not have a leading zero.
mm The minute. Single-digit minutes will have a leading zero.
s The second. Single-digit seconds will not have a leading zero.
ss The second. Single-digit seconds will have a leading zero
t The first character in the AM/PM designator defined in AMDesignator or PMDesignator, if any.
tt The AM/PM designator defined in AMDesignator or PMDesignator, if any.
z The time zone offset ("+" or "-" followed by the hour only). Single-digit hours will not have a leading zero. For example, Pacific Standard Time is "-8".
zz The time zone offset ("+" or "-" followed by the hour only). Single-digit hours will have a leading zero. For example, Pacific Standard Time is "-08".
zzz The full time zone offset ("+" or "-" followed by the hour and minutes). Single-digit hours and minutes will have leading zeros. For example, Pacific Standard Time is "-08:00".
: The default time separator defined in TimeSeparator.
/ The default date separator defined in DateSeparator.
% c Where c is a format pattern if used alone. The "%" character can be omitted if the format pattern is combined with literal characters or other format patterns.
\ **c** Where c is any character. Displays the character literally. To display the backslash character, use "\\".

MinRepeat and MaxRepeat Properties

MinRepeat and MaxRepeat properties can be used to validate segment looping and limit how many times segment can repeat within loop.
Important: Segment section in error report is displaying parent segment name and not actual segment that has repeat problems, and message section shows actual segment with repeat problems. Exception to this is situation when top level segment such as ISA has repeat problems. Then Segment section will be empty or show actual segment name.



Example of MaxRepeat error when input has 193 HLs and we limit accepted maximum repeat value of only 1.

DefaultValidation Property

Basic validation is performed on EDI X12 and EDIFACT enveloping segments if DefaultValidation property is set to true.

EDI X12 basic validation:

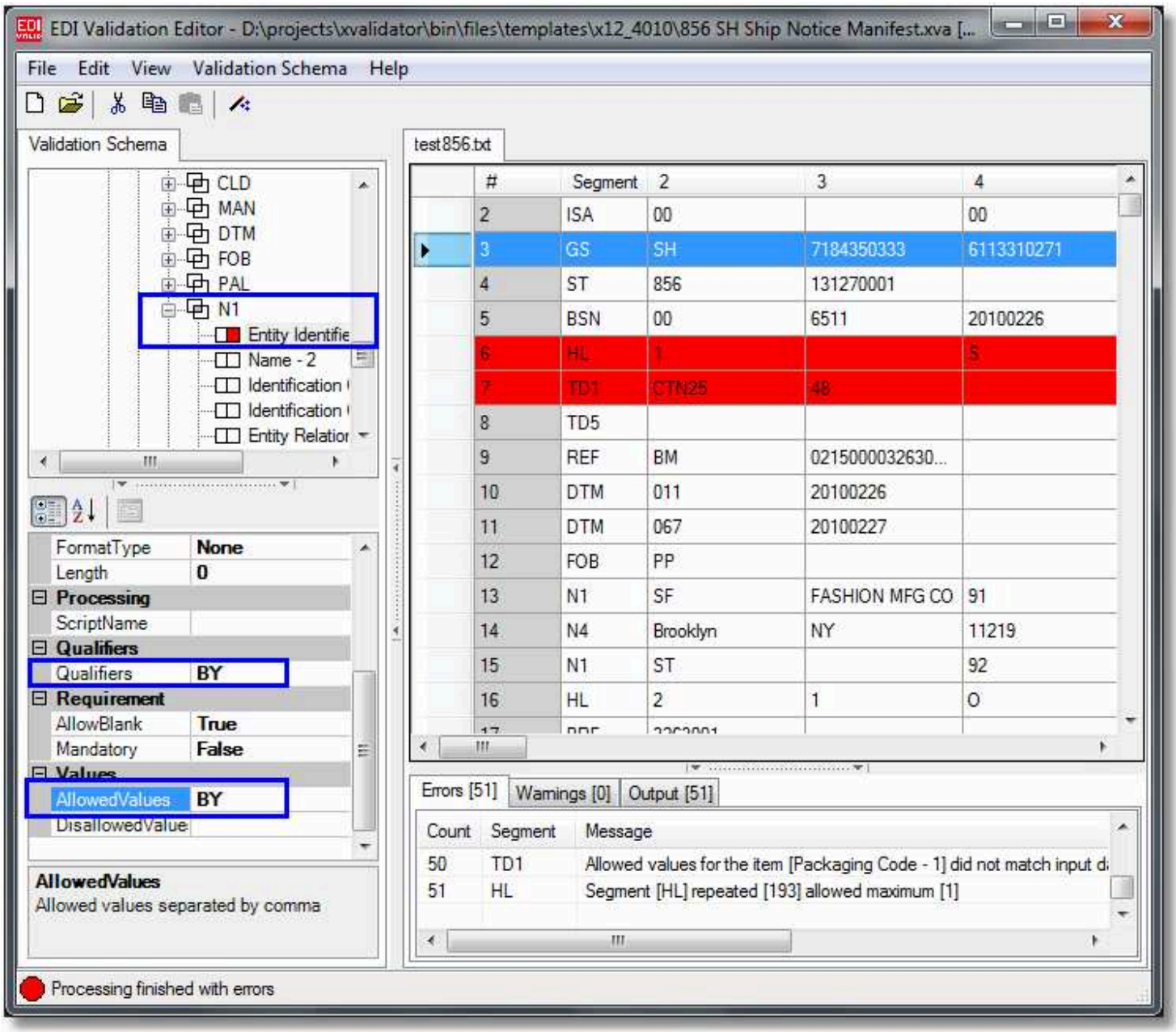
- Whether an ISA and an IEA exists.
- Whether ISA/08 contains a well-formed date value.
- Whether ISA/09 contains a well-formed time value.
- Whether ISA/13 contains legal boolean value.
- Whether ISA/14 contains legal interchange usage indicator (P, T, I).
- Whether ISA/12 and IEA/02 contain the same value.
- Whether IEA/01 contains the correct number of function groups in the interchange.
- If there is a matching GS and GE pair.
- Whether GS04 contains a well-formed date value.
- Whether GS05 contains a well-formed time value.
- Whether GS08 and GE02 contain the same value.

- Whether GS01 contains the correct number of messages in the function group
- If there is a matching ST and SE pair.
- Whether ST02 and SE02 contain same value.
- Whether SE01 contains the correct number of segments in the message.

Qualifiers Property

This property can be set on element. It allows you to match incoming segments not just by Tag property but also by qualifier values in the elements.
When validation process reads incoming segments it tries to match incoming segment to Tag properties of the segments in the schema definitions. If any segment in schema has elements with Qualifier property set, then validation process tries to match incoming element values to Qualifier property values set in the schema.
This is especially useful if you have few segments that have different validation rules based on qualifier values in them elements. Most typical ones in EDI X12 are N1, NM1, REF, DTM and PER segments that contain qualifiers.

You can enter few qualifiers in Qualifiers property separated by commas. Add same qualifier value to AllowedValues property.

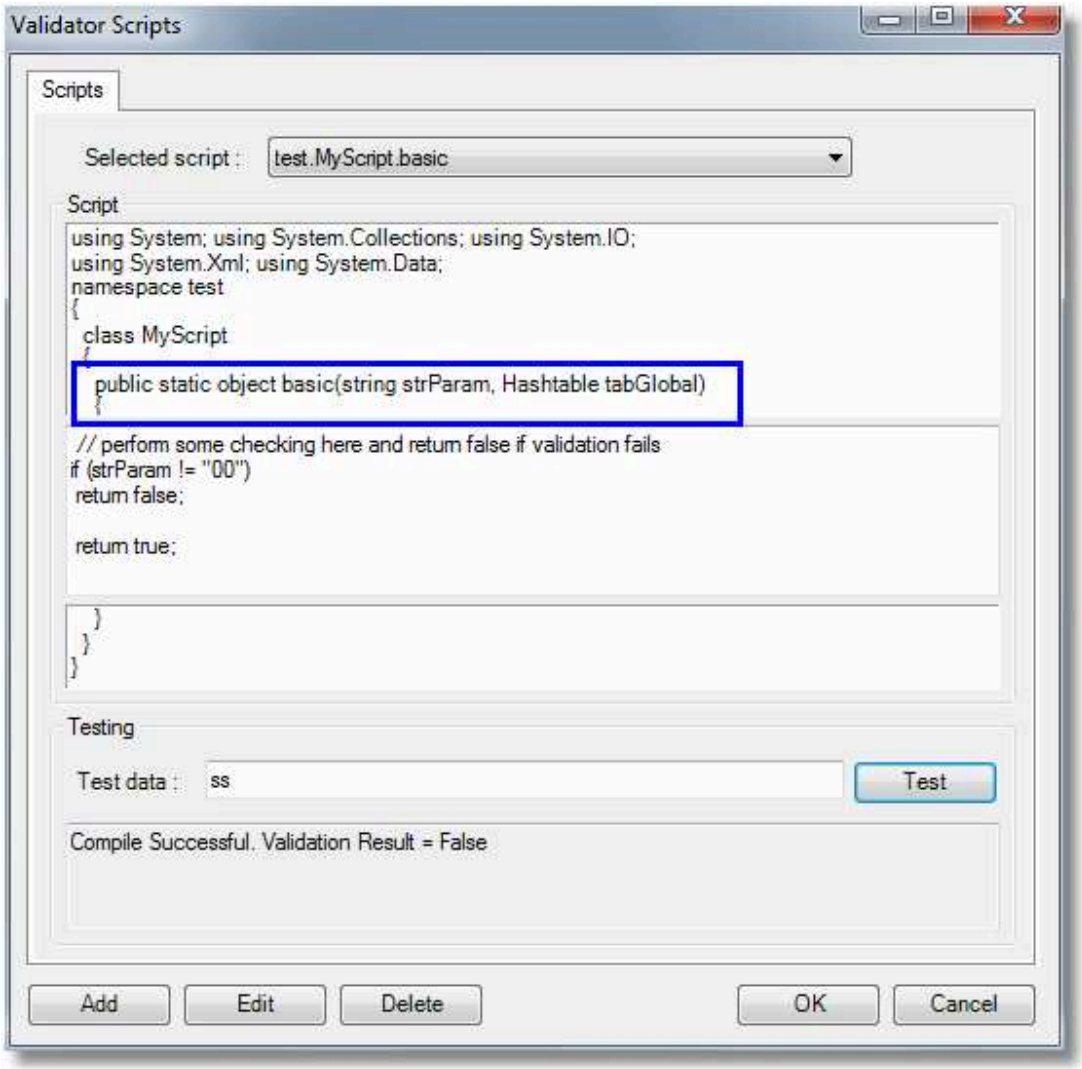


In this example N1 element 1 contains qualifier BY, and each N1 is separated into individual N1s.

Scripts

Scripts provide extra facility to validate data against specific business rules. They can be developed in C# and added to the validation process at various points of execution. They can be attached to any object in the validation tree and get executed once validation engine receives data for that item. For example: if you set script on segment ISA element 3, it will be called when ISA is received from the input message and element 3 is processed. Script will be loaded, compiled, and called with first parameter being input data for ISA element 3.

If some element does not receive data, script will not be called and executed for it. Example: let say you have script attached to REF element number 4. However your input data always comes with REF having only two elements. That way script is never called.



Sample script compares incoming data with “00”, if data is not equal to “00” validation fails.

Scripts also allow you to implement cross reference validation. Since instance of Hashtable tabGlobal is shared by all scripts you can save data value of strParam (current data) in tabGlobal and retrieve it later in other script. In that second script you can compare previous data values and make a decision. Simply return “false” if you want script to fail the validation.

Segments run scripts at the start of processing, when segment is being parsed and at the end of processing when segment and all its elements are already processed. Therefore there are two types of properties called ScriptName and PostScriptName. Script attached via ScriptName property will get called when segment processing starts. Script attached via PostScriptName property will get called when all the elements of that segment have already been processed.

PostScriptName is usually used to check inter-element cross reference rules. For that you would attach scripts to elements that need to be referenced. You can use tabGlobal to hold certain element values. Then in PostScriptName for that segment you can compare values saved in tabGlobal and perform validation.

Technical Support

Please contact technical support if you have any questions or concerns. Contact information is listed on the product and company websites. For most technical support questions we will request processing log.