



# **Universal Format Plug-in User's Guide**

---

Version 3.5

<b>UNIVERSAL</b> .....	<b>4</b>
<b>TERMINOLOGY</b> .....	<b>4</b>
<b>CREATING A UNIVERSAL FORMAT</b> .....	<b>6</b>
CREATING A UNIVERSAL FORMAT BASED ON AN EXISTING UNIVERSAL FORMAT .....	6
CREATING AN EMPTY UNIVERSAL MESSAGE FORMAT .....	9
<b>UNIVERSAL EXTERNAL MESSAGE UI</b> .....	<b>12</b>
<b>EXTERNAL FORMAT</b> .....	<b>13</b>
UNIVERSAL FORMAT OPTIONS .....	14
<i>General Settings</i> .....	14
<i>Default Settings for Text Encoded Fields</i> .....	16
<i>Default Settings for Binary Encoded Fields</i> .....	22
<i>Default Delimiter Settings</i> .....	23
ENTERING A UNIVERSAL MESSAGE .....	27
<i>Adding Fields and Specifying Properties</i> .....	28
Adding a Fixed Length field.....	33
Adding a Length Preceded Field.....	36
Adding a Delimited field .....	40
Encoding Types .....	43
<i>Adding Fillers and specifying properties</i> .....	54
<i>Adding a Sequence type section and specifying properties</i> .....	58
Adding a Fixed Instances Sequence .....	61
Adding a Variable Instances Sequence.....	62
Adding a Discriminated sequence .....	68
<i>Adding a Choice Type Section and Specifying Properties</i> .....	75
<i>Adding an All type Section and specifying properties</i> .....	83
A simple illustration of using <i>All</i> type section.....	88
REMOVING A FIELD/SECTION.....	90
<b>UNIVERSAL PLUG-IN PEEK FORMULA FUNCTIONS</b> .....	<b>90</b>
<b>SAVE SELECTION AS TEMPLATE / ADD TEMPLATE</b> .....	<b>91</b>
SAVE AS TEMPLATE.....	92
ADD TEMPLATE .....	93
<b>SAVING A UNIVERSAL MESSAGE FORMAT</b> .....	<b>93</b>
EXPORTING A UNIVERSAL MESSAGE FORMAT .....	96
IMPORTING A UNIVERSAL MESSAGE FORMAT .....	98
<b>IMPORTING COBOL COPY BOOK STRUCTURE</b> .....	<b>99</b>
<b>APPENDIX</b> .....	<b>105</b>



# Universal

Universal format is a powerful flat file parser/writer that handles fixed width, delimited and tagged messages. This format can be used in modeling complex messages. It supports different types of fields such as fixed length fields, variable length fields, delimited fields, tagged fields and filler fields. It also supports different types of encoding for fields such as Text, Binary and Packed Decimal.

## See Also:

[Terminology](#)

[Creating a Universal Format](#)

[Universal External Message UI](#)

[External Format](#)

[Universal Plug-in Peek Formula Functions](#)

[Save Selection As Template / Add Template](#)

[Saving a Universal Message Format](#)

[APPENDIX](#)

# Terminology

## Section

A section is a group of related information made up of one or more fields and/or sub-sections that are collectively referred to as child elements. The name of a section should be unique at a given level, i.e., there cannot be duplicate sections at the same level in the hierarchy. The Universal format supports creation of three types of Sections:

1. Sequence
2. Choice
3. All

Usage of the term 'section' in the context of Universal refers to any of these three possibilities.

## Sequence

Sequence indicates that child elements may occur in specified order. There cannot be duplicate sequences (sequences with same name) at the same level.

**Note:**

The entire message format (consisting of Header, Data and Trailer sections) that you design using the Designer is a sequence, i.e. the data should appear in that order. For e.g., the incoming/outgoing message cannot have Trailer data in the beginning followed by Header data. Likewise, the fields/sections added at the root level under the Header/Data/Trailer sections also form a sequence.

**Choice**

Choice indicates that only one child element may occur. In a Choice, all child elements have the same starting position. Identification of choice in an incoming message is based on uniqueness of each choice. Choice is similar to C language Union or COBOL Redefines, in the sense that only one of the child elements can occur. Choices may be padded to be of the same length. In a Choice section, all child elements except one must be either self-discriminating (For e.g. tagged) or should have explicit discriminator.

**All**

All indicates that child elements may occur in any order. It can be called an Unordered Section.

**Field**

A data field or simply a field represents a single item of information.

**Filler**

This is a special field. It is basically used to reserve some empty bytes in the message format for filling the same with data whenever necessary in the future. A Filler field cannot be mapped and will not appear in mapping and validation screens. Filler fields are exempt from name uniqueness. A Filler field is always of type String.

**Template**

Whole or part of a Universal message format can be saved as a template in some location and can be added in any other Universal message format.

**See Also:**

[Universal  
Creating a Universal Format](#)

## Creating a Universal Format

A Universal message format can be created in the Designer either from

Existing Universal message format, or  
Empty message format

When using an existing format, the set of sequences, fields, fillers, choices are available for the newly created format. The user has the option of using the format as it is or change the mandatory elements or add new elements as per his requirements. The existing Universal formats are available as XML files in the location `<installation dir>\config\Universal\messages`

When using an empty message format, the user has to add the section, field, filler, etc. as per the message specification. This happens when the format has to be entered the first time, after which the format can be saved and used to build other formats using the first option.

### See Also:

[Creating a Universal Format based on an existing Universal format](#)

[Creating an empty Universal message Format](#)

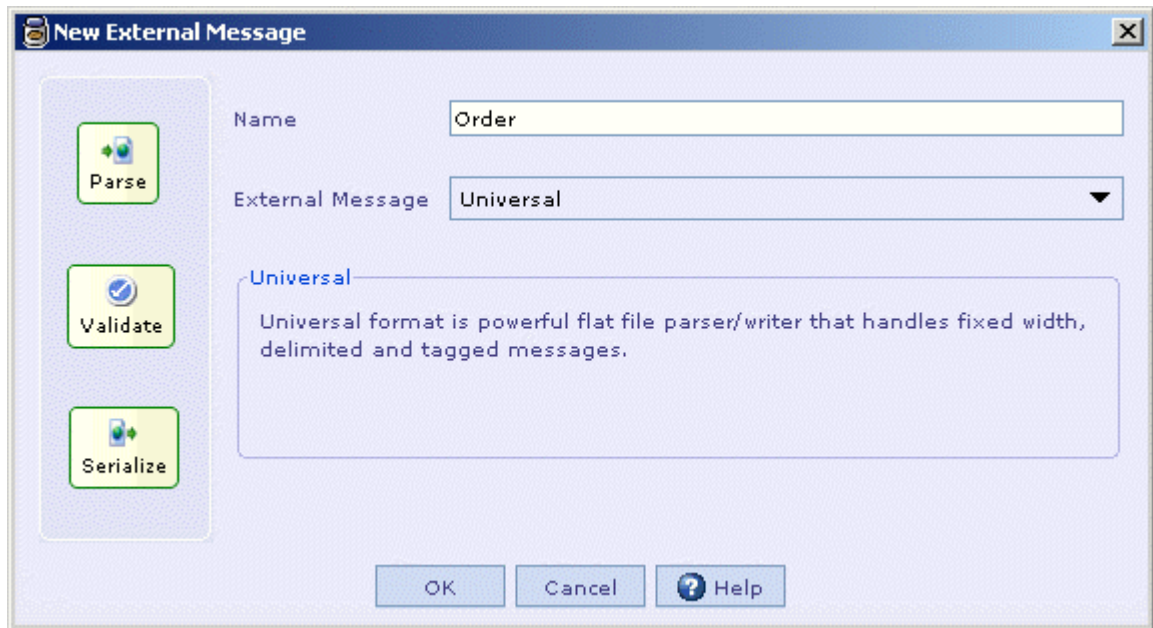
[Saving a Universal Message Format](#)

[Universal External Message UI](#)

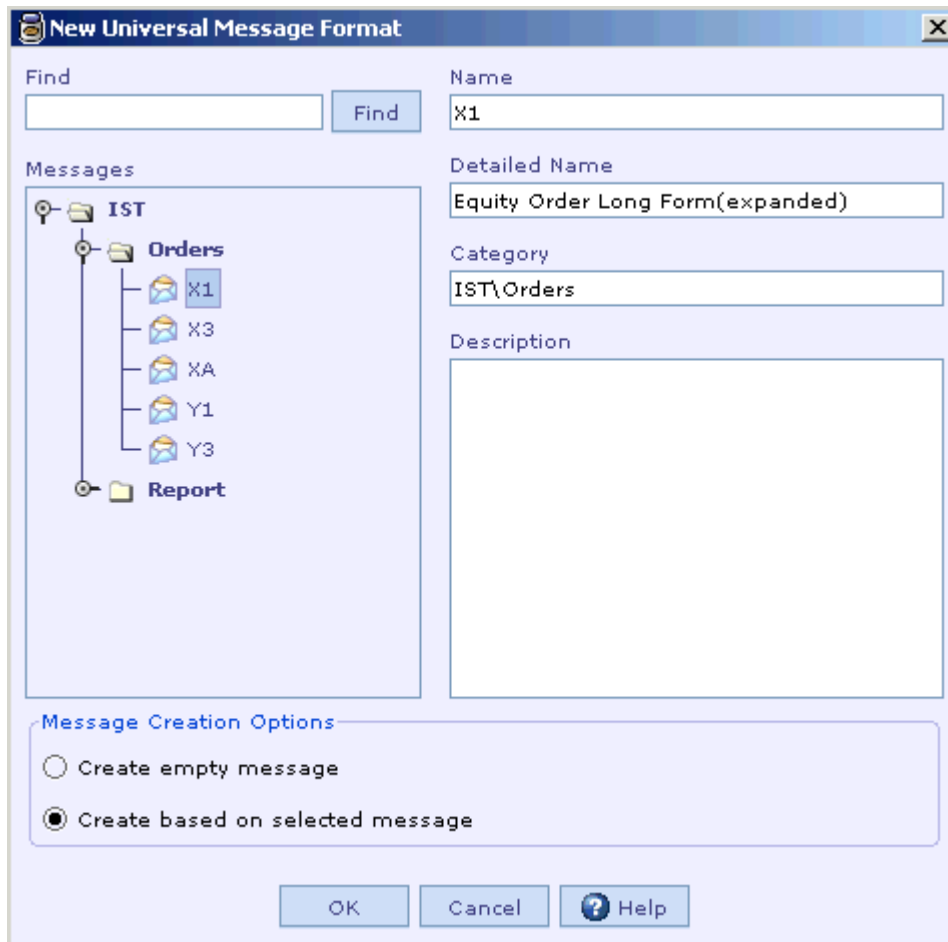
[External Format](#)

## Creating a Universal Format Based on an Existing Universal Format

1. Right-click the Cartridge node in the Designer and select the 'New External Message' menu item from the context menu to create a Universal external format.
2. In the 'New External Message' dialog that appears select 'Universal' from the 'External Message' list box and enter the message name in the 'Name' text field. Click OK.

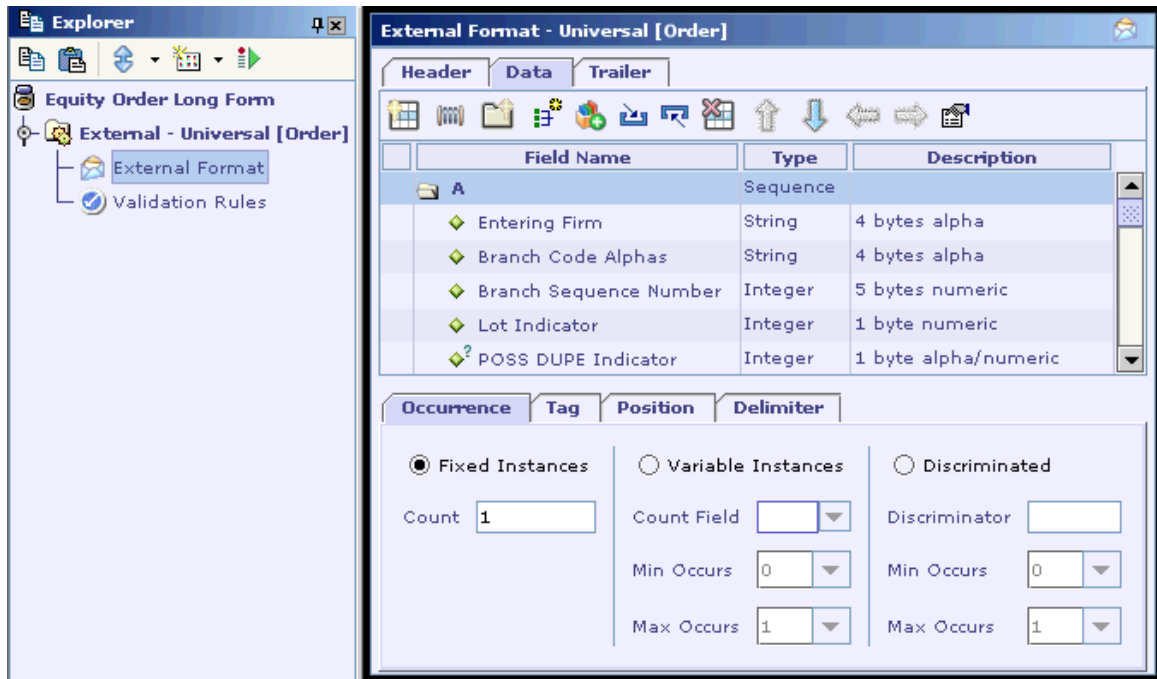


3. In the 'New Universal Message Format' dialog that appears, select an existing format based on which the new format is to be created.



4. As soon as you select an existing format, 'Create based on selected format' radio button gets automatically checked. Click OK.
5. The new format is created in the Designer as shown below.





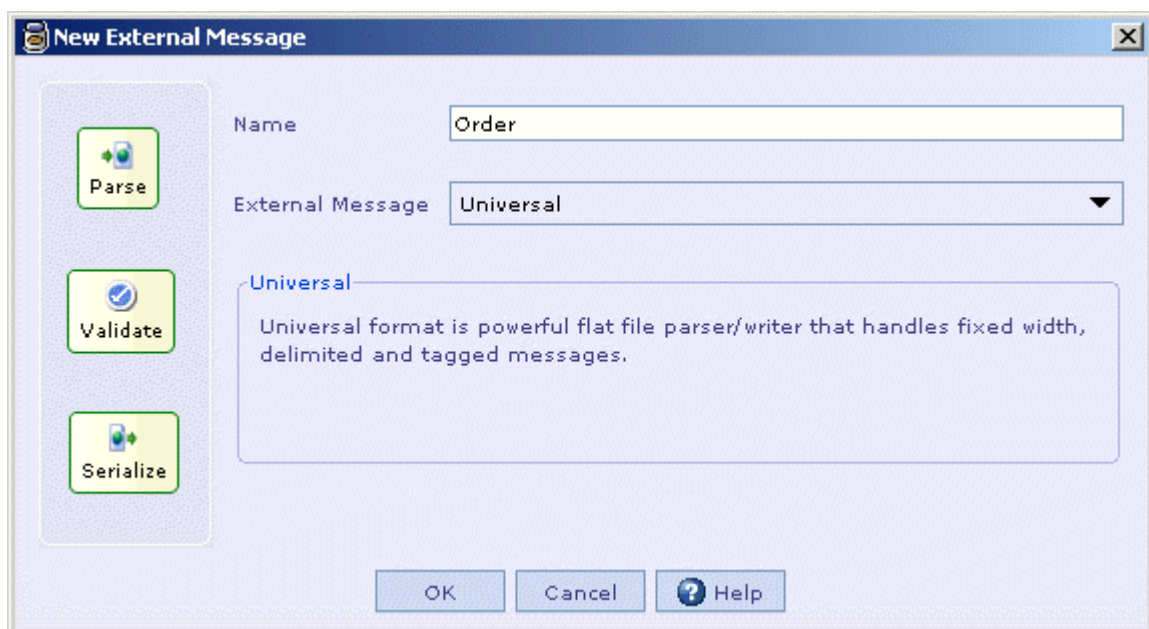
- Note that the Design Element UI tool bar buttons in the figure are enabled thereby allowing the user to modify the format.
- As seen in the above picture, three child nodes viz. 'External Format', 'Validation Rules' and 'Mapping Rules' with corresponding User Interface are created for a Universal External message format.

**See Also:**

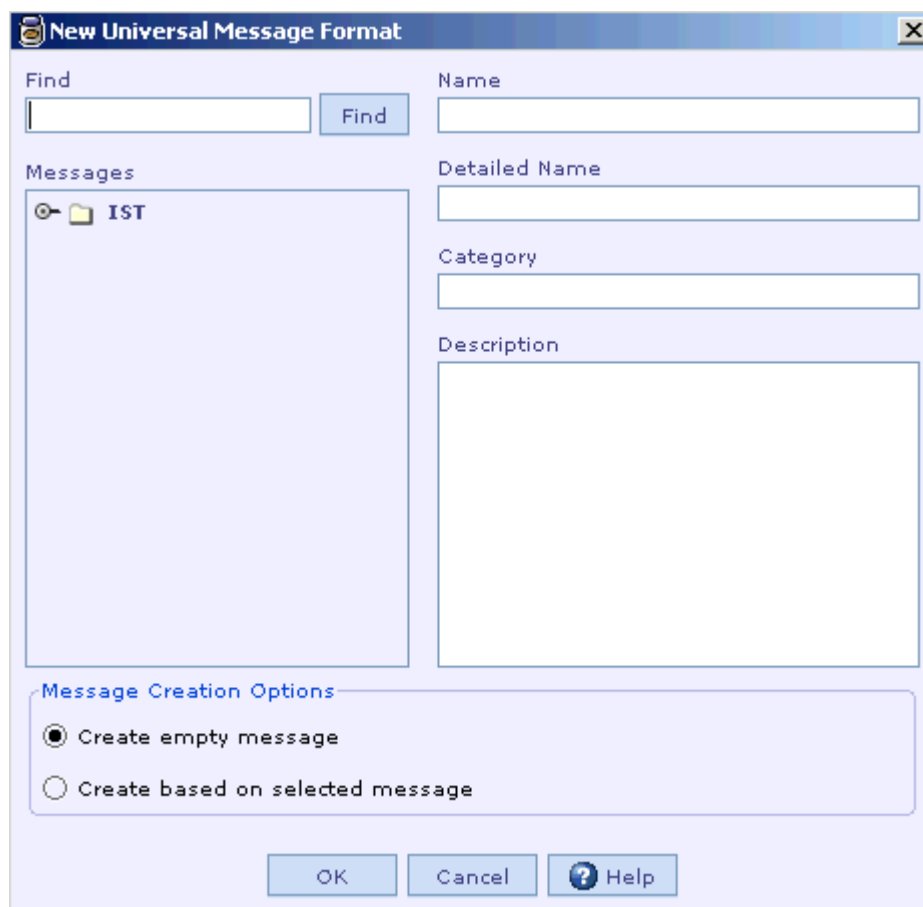
- [Creating an empty Universal message Format](#)
- [Saving a Universal Message Format](#)

## Creating an Empty Universal Message Format

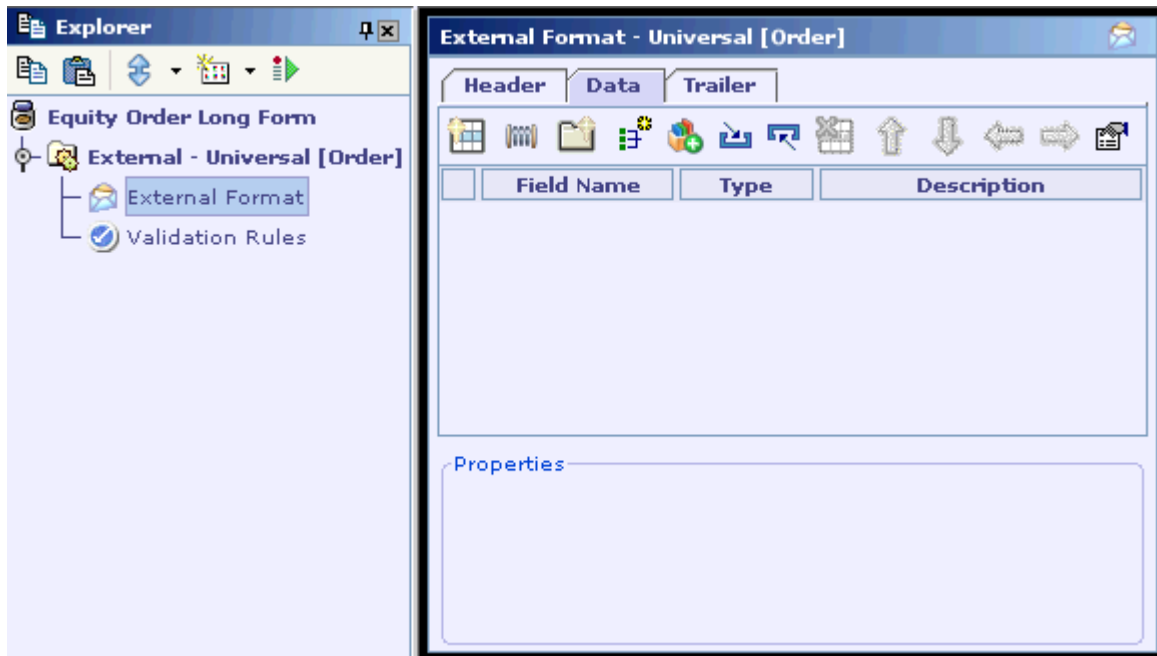
- Right-click the Cartridge node in the Designer and select the 'New External Message' menu item from the context menu to create a Universal external format.
- In the 'New External Message' dialog that appears select 'Universal' from the 'External Message' list box and enter the message name in the 'Name' text field. Click OK.



3. In the 'New Universal Message Format' dialog that appears, 'Create empty message format' radio button is selected by default. Click OK.



4. An empty message format is created as shown below.



5. In the Explorer window, the newly created Universal Message format is displayed with three child nodes viz., External Format, Validation Rules and Mapping Rules.

**See Also:**

[Creating a Universal Format based on an existing Universal format](#)  
[Saving a Universal Message Format](#)

## Universal External Message UI

After a universal external message has been created the following properties can be specified in the External Message UI.

- External Format Name
- Version
- Standard Name: This refers to the actual message name.
- Standard Version
- Detailed Name
- Description

**External - Universal [Equity Report]**

**Format Details**

External Format: Universal

Name: Equity Report

Version: 1.0

**Standard Details**

Name: A7

Version:

Detailed Name: ort Long Form (expanded order +Misc Text 3F/4)

Description:

Version Info ...

**See Also:**[External Format](#)[Creating a Universal Format](#)[Saving a Universal Message Format](#)


## External Format

The External Format User Interface is similar for Universal – Input / Output message formats. A Universal message consists of Header, Data and Trailer segments. The Header and Trailer segments are optional. The Header, Data and Trailer segments are represented by three tabs in the External Format UI. The same set of tool bars (properties for fields / sections) are available in the External Format UI under each of these three sections as shown below:

**See Also:**[Universal Format Options](#)[Entering a Universal message](#)[Creating a Universal Format](#)

## Universal Format Options

Before you start entering the actual message format, the Designer allows you to specify the default settings, which can be used with the fields and sections of the format. This setting is optional and is just for convenience. You can later change the properties as per your choice. Unless you explicitly set the field level properties, the format level default will be used.

Click on the 'Format Options' button  in the 'External Format UI' toolbar to bring up the 'Universal Format Options' dialog box that can be used for specifying the four format level options described below.

### See Also:

[General Settings](#)

[Default Settings for Text Encoded Fields](#)

[Default Settings for Binary Encoded Fields](#)

[Default Delimiter Settings](#)

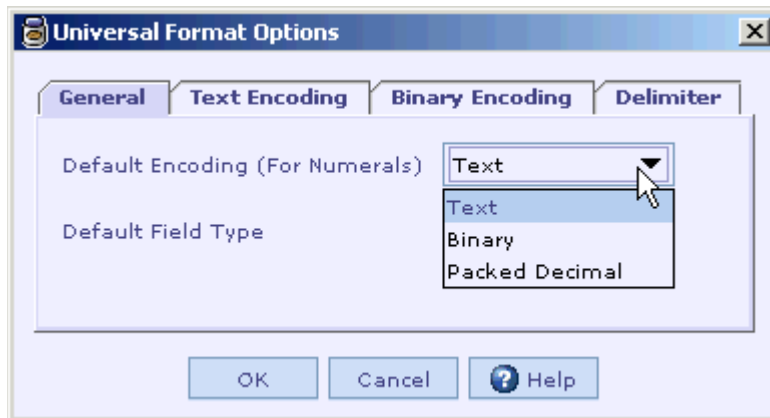
[Entering a Universal message](#)

## General Settings

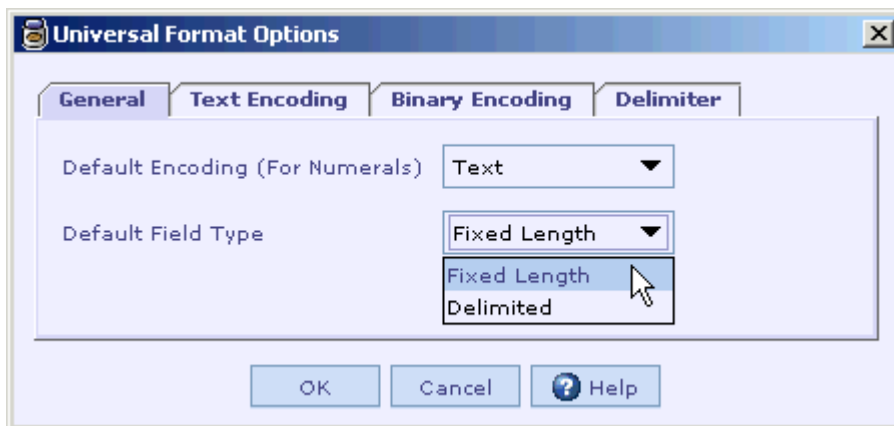
The 'General' tab of the 'Universal Format Options' dialog box is used to specify general settings described below. These property settings are applicable only for fields.

The 'Default Encoding (For Numerals)' list box is used to specify the default encoding to be used for fields of numeric data type such as Integer, Long, Float and Double. The user can select any one of the following encodings from the list box:

1. Text
2. Binary
3. Packed Decimal



The 'Default Field Type' list box is used to specify the field type to be used for fields by default. This is applicable for fields of all data types. The user can select between Fixed Length and Delimited field types.



Note that some of the combinations like 'Binary' or 'Packed Decimal' selected in 'Default Encoding (For Numerals)' list box and 'Delimited' selected in 'Default Field Type' list box are not meaningful. You will get a validation error in such cases if not set right at field level.

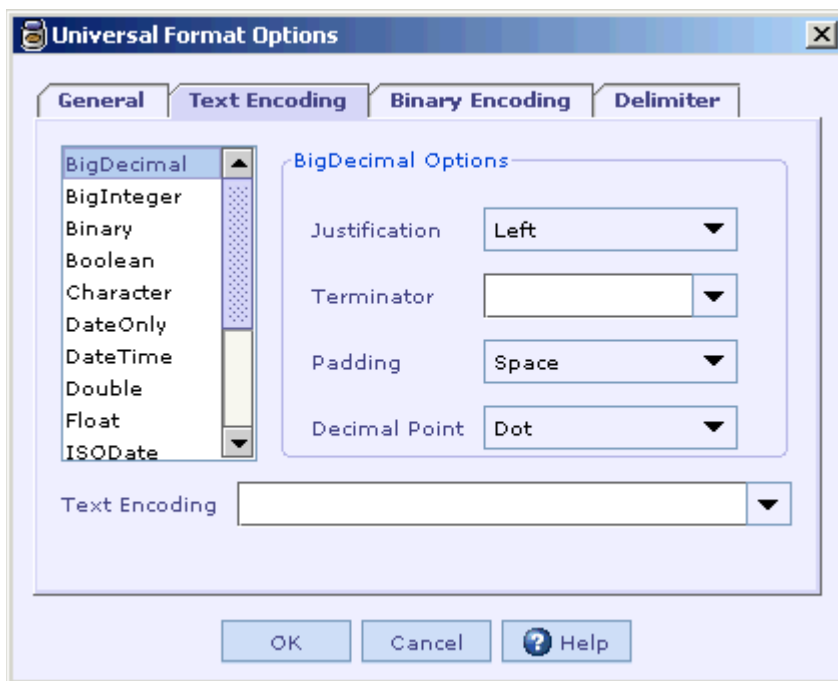
**See Also:**

- [Default Settings for Text Encoded Fields](#)
- [Default Settings for Binary Encoded Fields](#)
- [Default Delimiter Settings](#)
- [Entering a Universal message](#)

## Default Settings for Text Encoded Fields

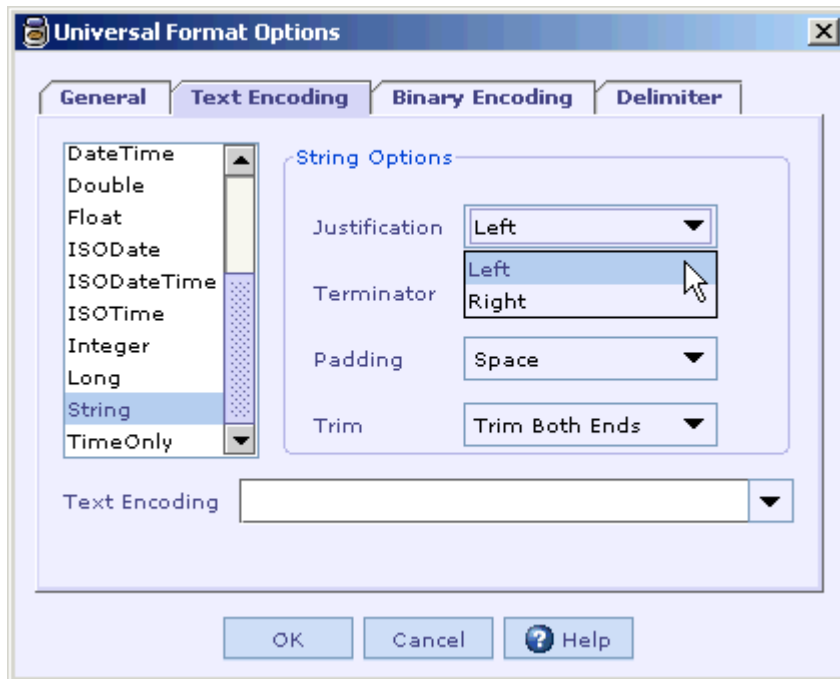
The 'Text Encoding' tab of the 'Universal Format Options' dialog box can be used to specify the default settings to be used with Text encoded fields of different data types. This property is applicable only for fields. If the corresponding value at the field level is set to 'Default', the field *inherits* values set at the format level. You can override these attributes at the field level by specifying a value other than 'Default'.

As can be seen in the picture given below, the data types are listed on the left side of the 'Universal Format Options' dialog when 'Text Encoding' tab is selected. On selecting a particular data type, the options specific to that data type are shown on the right side of the dialog. These options have to be specified for each data type.



The **Justification** list box is used to specify the default justification to be used during output transformation when the value of the field is less than its specified width. The user can select between Left and Right justifications.





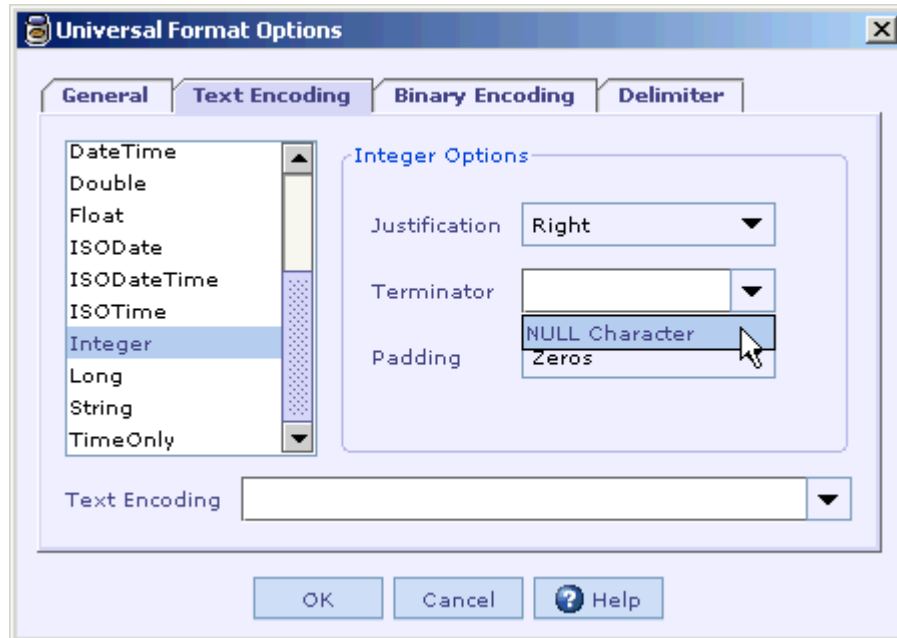
For e.g., consider a Universal Output Message format containing an integer field of fixed length 5, which is Right justified, and Padding option selected as 'Zeros'. If value of the source field in the Input Message is "23", on message transformation, the value of this target field would be "00023".

**Note:**

There is a relationship between the 'Justification' and 'Padding' properties in case of numeric fields of data types Integer, Long and BigInteger. If Justification for a field of one of the above types is selected as 'Left', then the user should compulsorily select 'Space' as padding option from the Padding list box. On the other hand, if Justification for the fields of the above mentioned types are selected as 'Right', then either 'Space' or 'Zeros' padding can be given.

The **Terminator** combo box is used to specify the default terminator to be used to indicate the end of the field value. For e.g., if a field is of fixed length 10 and if you give terminator as, say comma (,) then, the value of the field that is available till comma is encountered, will be taken as the actual value and the rest of the values after comma would be ignored. Terminator should be a single character (can be alphabet, numeric or special character; typically the NULL character) and can be specified for Fixed Length, Length Preceded and Delimited fields. In case of delimited fields, make sure that the Terminator does not conflict with the Delimiter. Refer [APPENDIX](#) to find out the various escape sequences that can be specified for a Terminator. (The difference between a

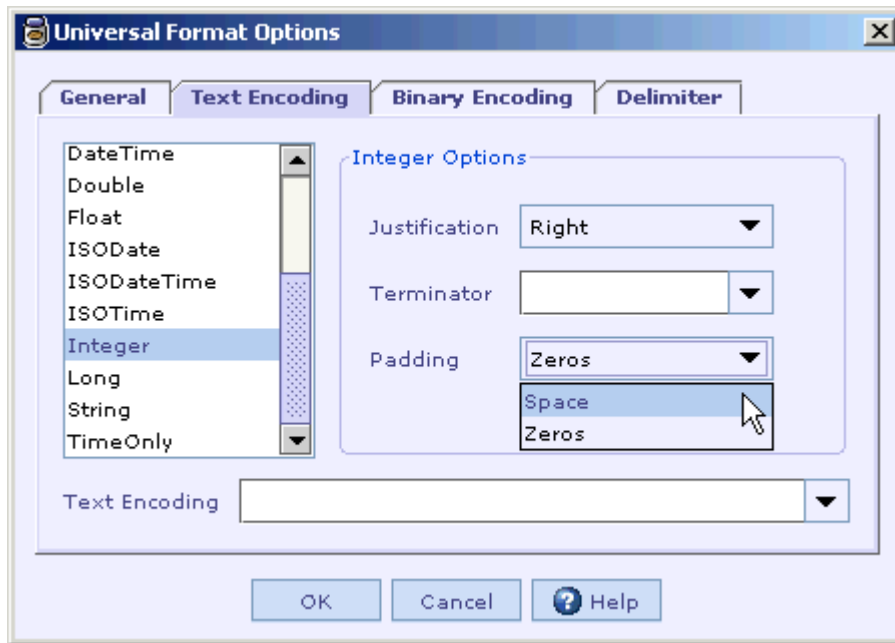
Terminator and a Delimiter is that while the 'Delimiter' denotes the end of the field width, the 'Terminator' denotes the end of the field value.)



**Note:**

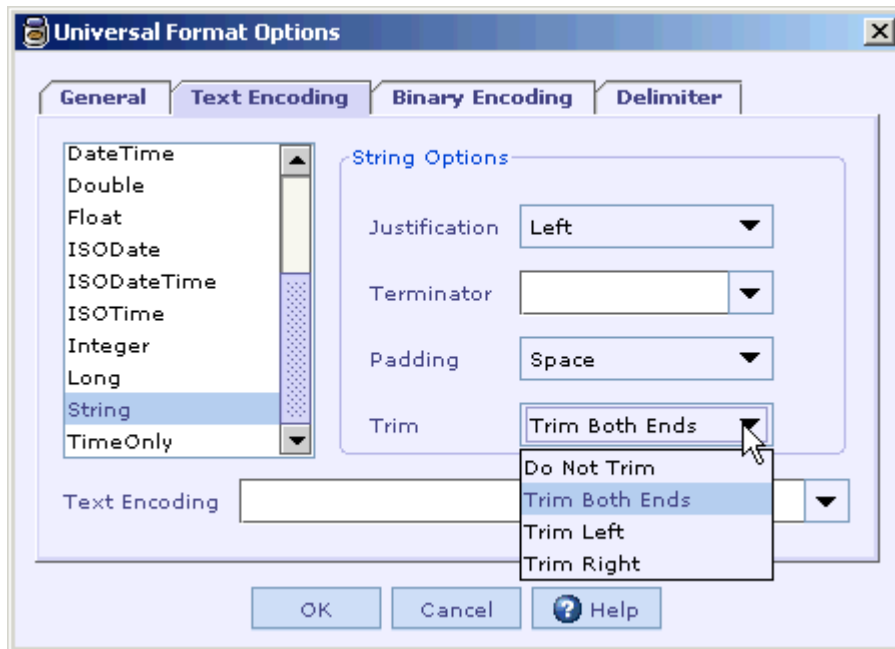
The NULL Character terminator given in the combo box would be helpful in case the user wants to enter a "C" structure as a message format.

The **Padding** list box (applicable for all data types except Date) is used to specify the default padding to be used to fill the balance width when the value of the field width is less than its pre-specified width. For fields of data types String and Boolean, only one padding option is supported namely 'Space'. For fields of Integer, Float, Double, Long, BigInteger and BigDecimal data type, two padding options are supported namely 'Space' and 'Zeros'. However, if fields of Integer, Long and BigInteger data type are left-justified, 'Zeros' padding should not be specified as noted above.

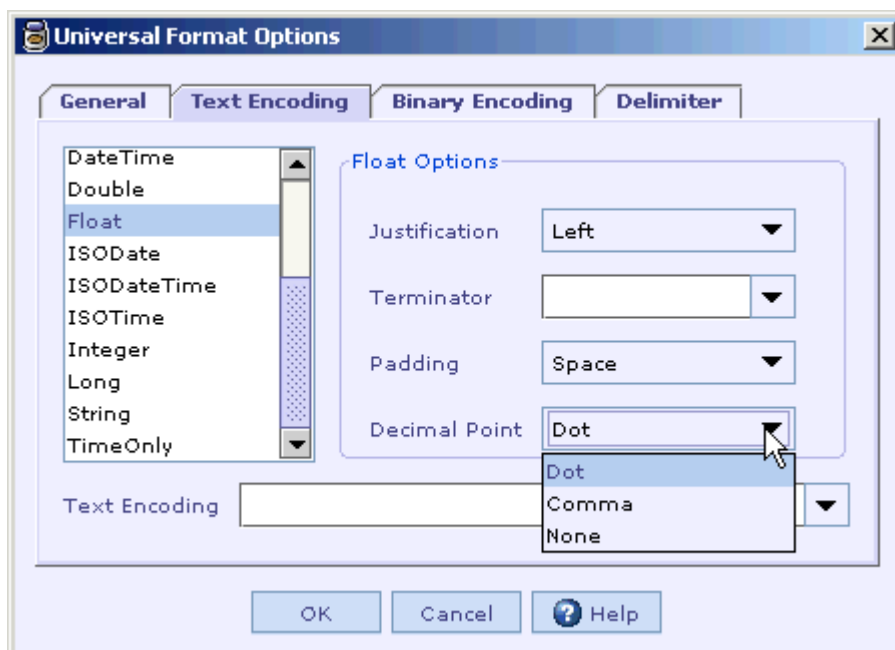


The **Trim** listbox (applicable only for string and Character data type) is used to specify if the value of the Text encoded field has to be trimmed during input transformation (ignore padding, if any in the field) or not and how it has to be trimmed. The user can select between the four options available namely Do Not Trim, Trim Both Ends, Trim Left and Trim Right. This property can be used for Fixed Length, Length Preceded as well as Delimited Fields. For e.g. consider a Fixed Length string field of length 10. Let its value in the input message be " AB " (padding of 4 spaces given at both ends). Then the actual value of the field on parsing depends on the Trim option chosen as shown in the following table:

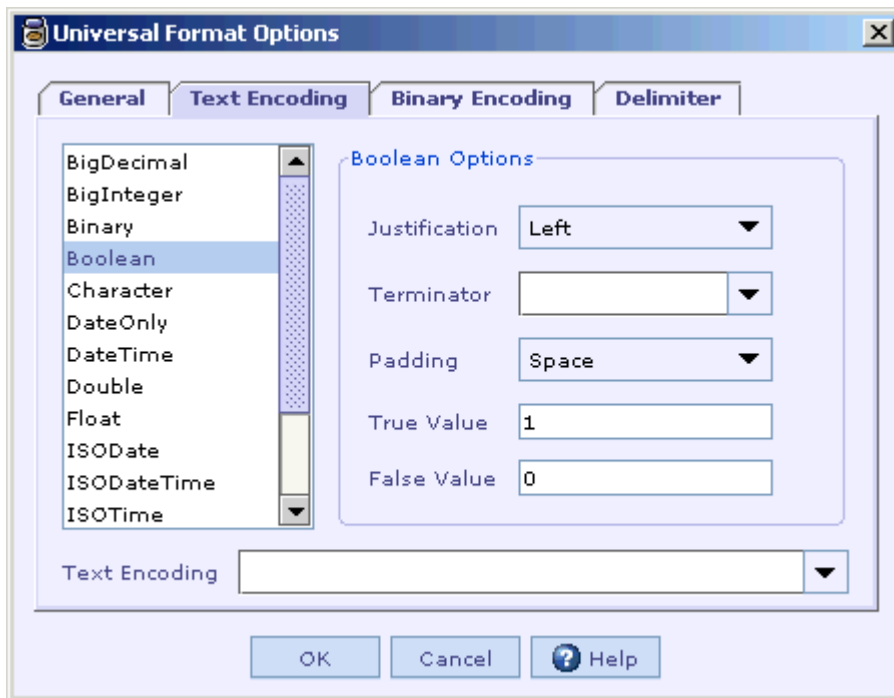
S.No.	Trim Option Chosen	Actual value on parsing
1	Do Not Trim	" AB "
2	Trim Both Ends	"AB"
3	Trim Left	"AB "
4	Trim Right	" AB"



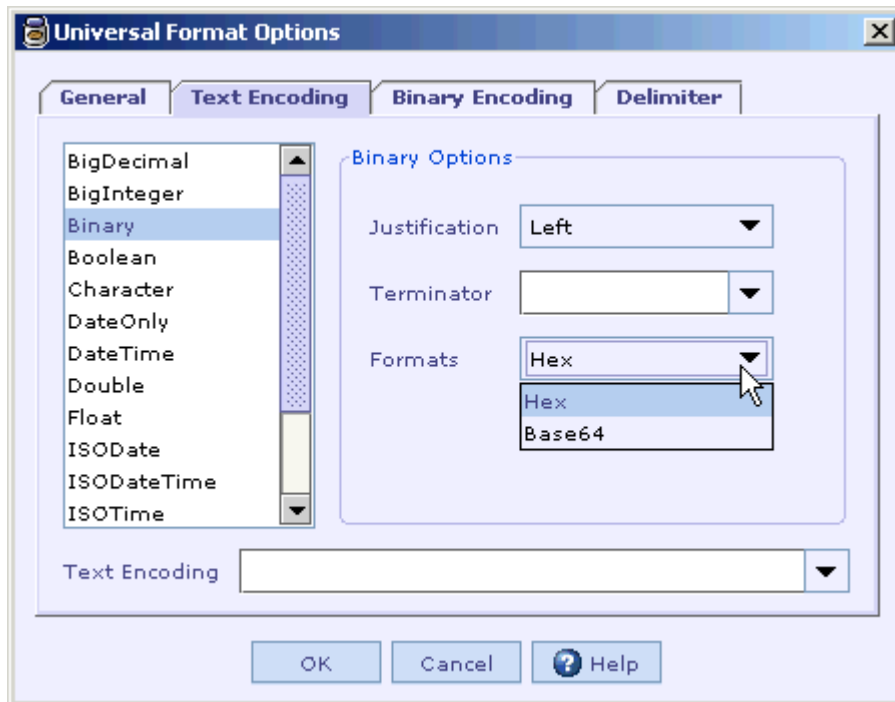
The **Decimal Point** listbox (applicable only for Float, Double and BigDecimal data types) is used to specify the character to be used to denote a decimal point. The user can select between the three options available namely Dot, Comma and None. If the option None is selected, then the decimal point need not be given in the input message (implied) and the real and fractional part of the field are determined by the specified format.



The 'True Value' and 'False Value' text fields (applicable only for Boolean data type) are used to specify the values corresponding to Boolean values **true** and **false**. This property is mandatory. Note that by default 1 is used for representing **true** and 0 is used for representing **false**. If any of these text fields is left empty, it will result in validation error.



The **Formats** list box (applicable only for Binary data type) is used to select between the binary formats 'Hex' and 'Base64'.



The '**Text Encoding**' combo box is used to specify the character encoding to be used while processing text encoded fields in the message. The default character encoding used for text-encoded fields is 'US-ASCII'.

**See Also:**

[General Settings](#)

[Default Settings for Binary Encoded Fields](#)

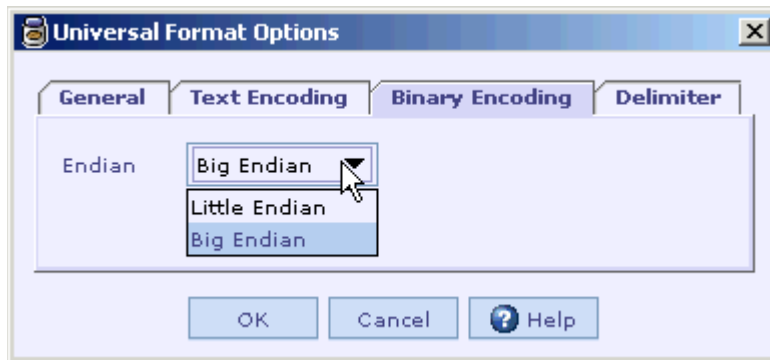
[Default Delimiter Settings](#)

[Entering a Universal message](#)

## Default Settings for Binary Encoded Fields

The **Binary Encoding** tab of the 'Universal Format Options' dialog box can be used to set the default endian for a Binary Encoded field, i.e. the byte order specific to the Microprocessor.

The **Endian** listbox is used to specify the Endian to be used for a Binary Encoded field. The user can select between the two options available namely Little Endian and Big Endian.

**See Also:**[General Settings](#)[Default Settings for Text Encoded Fields](#)[Default Delimiter Settings](#)[Entering a Universal message](#)

## Default Delimiter Settings

### Header / Data / Trailer Delimiter

Message is a sequence of Header, Data and Trailer. As discussed earlier, Header and Trailer may be optional. When Header and Trailer fields are specified in the message format, and if a new-line has to separate the Header from Data or the Data from the Trailer, it has to be explicitly specified as `\r\n` in "Delimiter" combo box. (If no fields are specified in Header or Trailer segments of the External Format UI, then the first and last records should be treated as belonging to the message body)

The Delimiter combo box provided under **Header Delimiter**, **Data Delimiter** and **Trailer Delimiter** is used to specify the delimiter to be used for Header, Data and Trailer segments. These are not the default but the **actual** values. That is why in the corresponding "Universal Format Options" UI, you don't find the word "Default" mentioned when these delimiter combo-boxes are referred.

**Note:**

If you specify a delimiter for, say Header, then the value of any of the fields in the Header should not contain the specified delimiter, the delimited sections and the delimited fields included in the Header should not have the same delimiter as that of the Header, because the parser assumes that the Header ends when the delimiter is encountered.

### Record Delimiter

The Delimiter combo box provided under **Default Record Delimiter** is used to specify how Records (Sections) are separated from each other or how different instances of a record may be separated from one another. The default delimiter you specify here is used, provided you haven't overridden it at section level. Records may be delimited by "\n" or "\r\n" or any other specified literal. Special characters are represented as "\#xH", where H represents a hex number which is a Unicode number. For example, the form feed character (Unicode #xC) can be represented by "\#xC". The line feed character (Unicode #xA) can be represented as "\#xA". Delimiter is optional for records. Refer [APPENDIX](#) to find out the various escape sequences that can be specified for a Delimiter.

**Note:**

If you specify a delimiter for a section, then the value of any of the fields in the section should not contain the specified delimiter and the delimited fields included in that section should not have the same delimiter as that of the section, because the parser assumes that the section ends when the delimiter is encountered.

**Field Delimiter**

The Delimiter combo box provided under **Default Field Delimiter** is used to specify the default delimiter to be used to separate Fields within the Section. The field delimiter is used for delimited fields alone, i.e. only if you have selected the Field Type as **Delimited**. This attribute can be overridden at the field or section level. (If a delimiter is set at field level, that is used, if not, the default set at the section level is used, if not, one at the format level is used).

**Note:**

1. The last field in a section is terminated by section delimiter and field delimiter is omitted. The last section added at root level (not followed by any other field/section) in the Header/Data/Trailer is terminated by Header/Data/Trailer delimiter, if any specified and section delimiter is omitted, i.e., section delimiter need not be given for the same in the message.
2. A delimiter can be of any length and can be a combination of both alphanumeric and special character.

**Quote**

The Quote combo box provided under **Default Field Delimiter** is used to specify the escape sequence to be used in case the delimiter character/s is/are to be included in the value of the field. Refer [Quote under Adding a Delimited field](#) for more details.



## Ignore Trailing Fields

**Ignore Trailing Fields** checkbox can be checked if the optional trailing fields of a **delimited** Header / Data / Trailer / Section (Record) are to be ignored. This property is used for convenience to avoid giving empty field values (spaces) in case the optional trailing fields are of null value. If "Ignore Trailing Fields" check box is not checked, wherever the optional trailing field is of null value, empty field values (spaces) should compulsorily be given for optional field, which has null value. It is to be noted that this option is applicable and is of use only if the following are satisfied:

The section should be a delimited one.

The trailing fields are set as optional, are of null value and no default value given for the same. (The trailing fields may be of Fixed Length, Length Preceded or Delimited)

### Note

There is a difference in the application of default settings to the message format when specified under the "General" tab and under the other tabs of the "Universal Format Options" window.

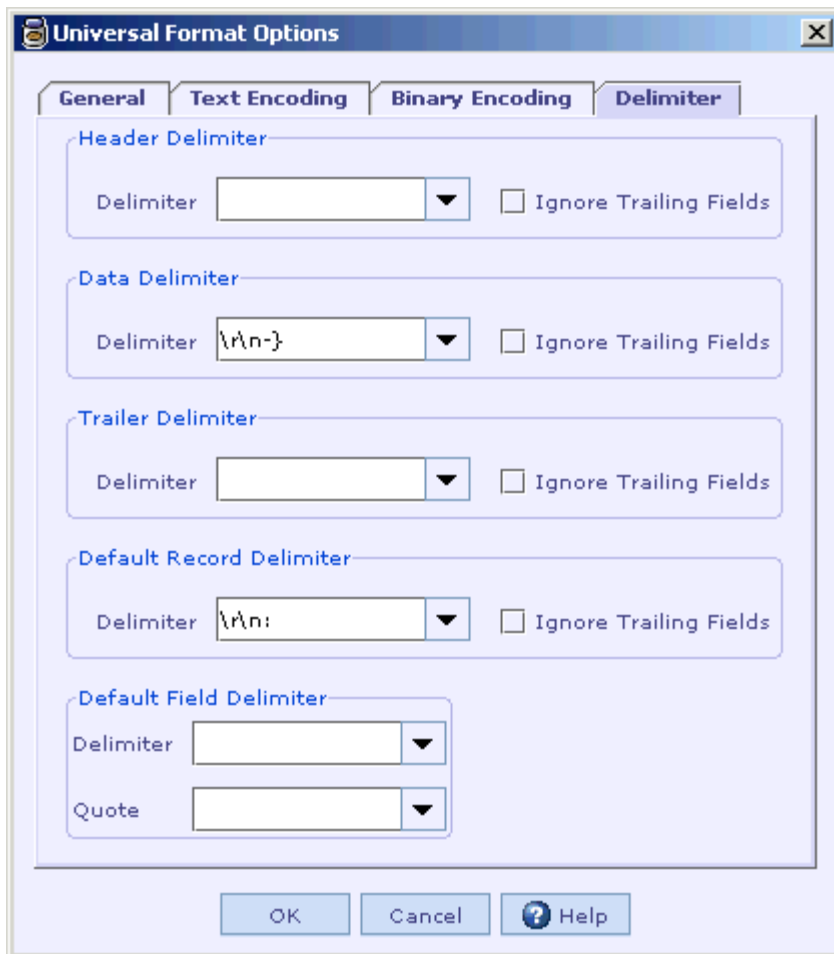
When we set various default settings available under the tab "General", they are applied as well as displayed as selected, to any field added subsequently, and not applied to the fields which were added before giving the default settings using the Format Options feature.

For e.g. consider an integer field with name A added in the External Format with the Encoding under the "General" property chosen as Text. Go to the "Universal Format Options" window and under Default Encoding (For Numerals) list box of the "General" tab, select Binary. Now add another field of integer type with name B in the External Format. You would find the option Binary automatically selected for field B in the Encoding list box under the "General" tab in the Properties panel, while the Text encoding selected for field A remains as such.

In case of default settings selected under other tabs viz., "Text Encoding", "Binary Encoding" and "Delimiter", they are not displayed as selected, but displayed as "Default" in the corresponding place and also, the value in "Default" is inherited even by the previously added fields whenever a change is made subsequently using the "Format Options" feature.

For e.g., before adding any field in the External Format, go to the "Universal Format Options" window and under the tab "Delimiter", select Default Field Delimiter as comma (,) and Quote as double-quotes ("). Now add a string field in the External Format and select Field Type as "Delimited". Under the Delimiter

and Quote combo-boxes, you would find "Default" selected automatically (Note that though you had selected comma in the Format Options window, it is not displayed as such, but only displayed as Default, meaning thereby that the value of Default can be changed again using the Format Options feature). Here the "Default" selected in the Delimiter combo box would mean comma while the "Default" selected in the Quote combo box would mean a quote. This can be easily verified by giving data, say "abcde,fg", in the Simulator tool. On parsing you would get the value of the field as abcde,fg. Now go to the "Universal Format Options" window and select Default Field Delimiter as semi-colon (;) and Quote as single-quote ('). Now in the External Format, the meaning of "Default" selected in the Delimiter and Quote combo boxes would have changed as set under Format Options, this again can be verified by giving the same input data as above in the Simulator tool, where, on parsing you get some undesired output. If you give input data as 'abcde;fg'; on parsing you would get the desired output namely abcde;fg.



**See Also:**

[General Settings](#)

[Default Settings for Text Encoded Fields](#)

[Default Settings for Binary Encoded Fields](#)

[Entering a Universal message](#)

## Entering a Universal Message

A message can be entered using Universal Format by adding various combinations of Fields, Fillers, Sequences, Choices and All.

**See Also:**

[Adding Fields and specifying properties](#)

[Adding Fillers and specifying properties](#)

[Adding a Sequence type section and specifying properties](#)

[Adding a Choice type Section and specifying properties](#)


[Adding an All type Section and specifying properties](#)

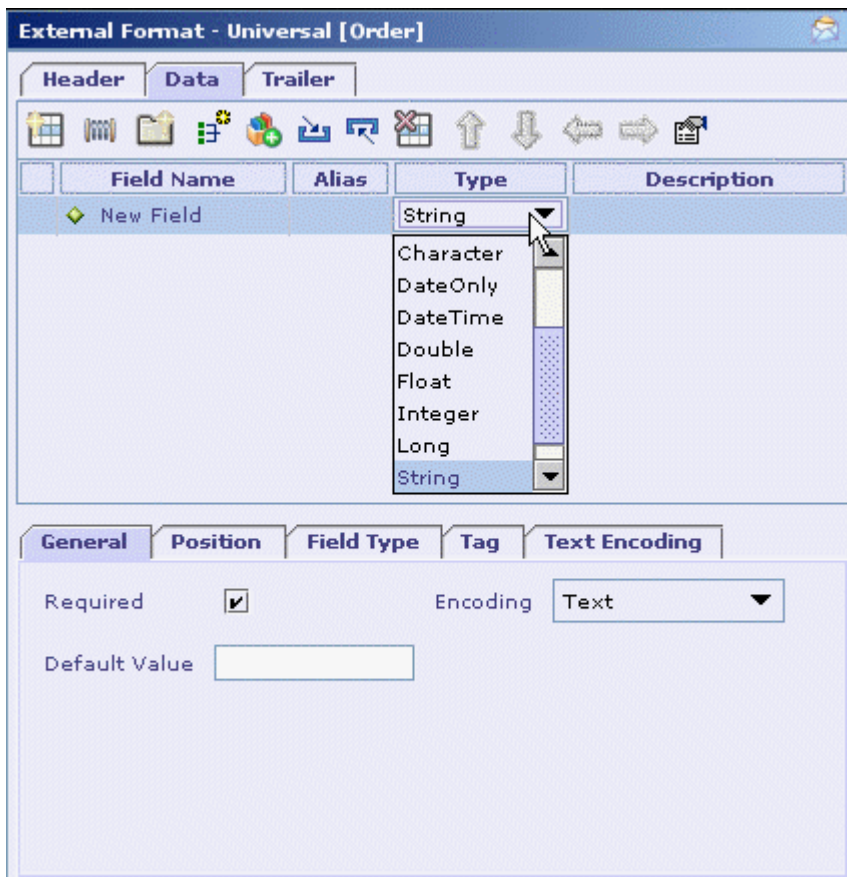
[Universal Format Options](#)

## Adding Fields and Specifying Properties

Universal Format supports adding fields of any of the thirteen data types viz. String, Integer, Character, Float, Double, DateTime, Long, Boolean, Binary, BigDecimal, BigInteger, TimeOnly and DateOnly. A wide range of properties can be specified for a field as discussed below:

### To add a field in the Universal - External Format UI

1. Click the **Add New Field** button  in the Universal - External Format UI.
2. A new row (field) is added in the External Format UI with default name 'New Field' and default type set as 'String' as shown below.



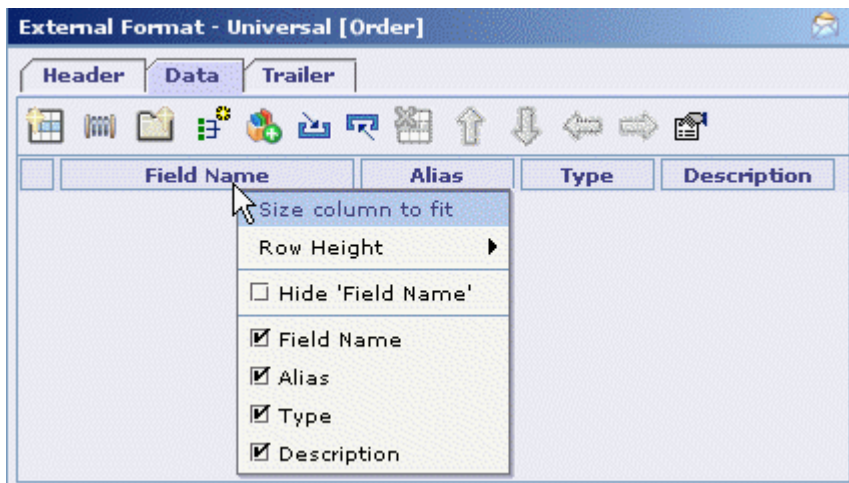
You can change the field name as per your requirement (A field should not have the name as "FILLER") by double-clicking on the column under "Field Name" and selecting any of the data types from the list box, which is enabled when you click

the **Type** column cell of the added row. Note that the message format is shown in the table at the top, and a panel at the bottom shows the properties of the row selected in the table. The properties panel dynamically changes as the row selected in the table changes.

- Under the column **Alias**, you can specify the alias (substitute name) to be used for the field when specifying a Validation/Mapping formula involving that field. This property is optional. (If a field is assigned an alias, the qualified name of that field is replaced with its newly assigned alias in all the places it is used). This column is hidden by default when a Universal message format is created and can be viewed by right-clicking the Table heading in the External Format UI and checking the Alias check-box in the context menu as shown in the picture added below.
- Under the column **Description**, you can mention any description for the added field. This property is also optional.

**Note:**

If you right-click the heading column of the table in the External Format UI, you get a context menu to hide columns of the table and also set properties for the table as shown in the following picture:



### Specifying Properties for a Field

The field properties panel has five tabs,

Property Tab	Usage
--------------	-------

General	Can specify the general properties of the field. Here you also set the encoding to be used for the field
Position	Displays the position of the field in the message, data and its length
Field Type	Can select the type of the field viz., Fixed length, Length preceded or Delimited.
Tag	Can specify the tag and separator for the field
Encoding	This tab changes based on the encoding selected in the General tab. As the name implies, displays encoding specific attributes of the field.

**General tab:**

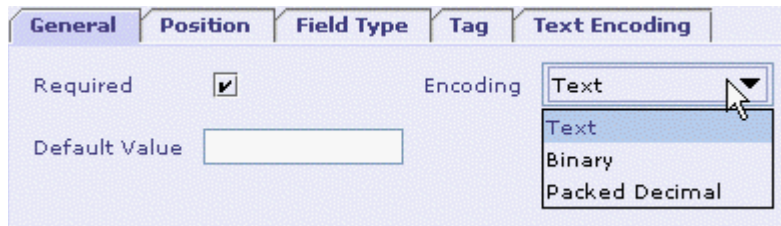
Under this tab, the properties that can be specified for a selected field are discussed as under:



**Required** check box is used to specify if the selected field is mandatory or not.

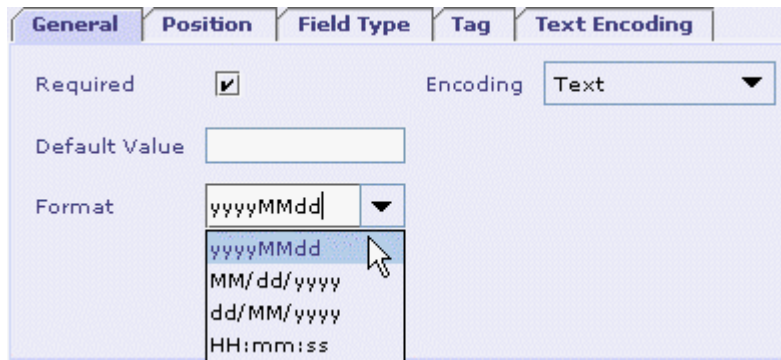
**Default Value** text-box is used to specify the default value of the field if any. This property is optional. Refer [APPENDIX](#) to find out the various escape sequences that can be used while specifying the Default Value.

**Encoding** list-box is used to specify the required encoding for the selected field. If the selected field is of String, Character, Date/Time, BigInteger or BigDecimal data type, the **Encoding** list-box has only one element viz., Text. However, if the selected field is of numeric data type (Integer, Float, Double or Long), the **Encoding** list-box has three elements (as shown in the following picture) viz., 'Text', 'Binary' and 'Packed Decimal'. Note that the name of the 'Encoding' tab changes dynamically based on the encoding chosen in the 'General' tab, i.e., if the current encoding chosen is Text and if we change it by selecting 'Packed Decimal' from the 'Encoding' list box, then the name of Encoding tab changes from 'Text Encoding' to 'Packed Decimal Encoding'.



If the data type of the selected field is 'Binary' or 'Boolean', the Encoding list-box displays two elements representing the two different encodings applicable viz., Text and Binary.

If the selected field is of Date/Time data type, its pattern can be specified in the 'Format' combo box that is displayed as shown in the following picture. This property is mandatory.



**Note:**

The date/time pattern to be specified is case-sensitive and the valid usage is as under:

yyyy or yy	year
MM	month
dd	days
HH	hour
mm	minutes
ss	seconds
SSS	milli seconds

**Position tab:**

Under this tab, the position of the selected field and its length are displayed. The position counting starts from zero. The values displayed under this tab are read-only and they change dynamically according to the position of the selected field. This property is generally of use if the message entered has fixed length fields. Under this tab, the following properties are displayed for a selected field:

**Position in Message** which is determined by the position of the selected field in the Message, i.e. including Header.

**Position in Data** (if the selected field is in the Data section) which is determined by the position of the selected field in the Data. Note that if there is no Header at all for the message format, then this value and the value against **Position in Message** would be one and the same.

**Length** which is determined by the length of the field specified under the Field Type tab. This is applicable only in case of Fixed Length fields.

General	Position	Field Type	Tag	Text Encoding
<b>Field Position</b>				
Position in Message	6			
Position in Data	6			
Length	1			

Note that the value specified for Length includes the length of the Tag if any specified for the field. For e.g., if a field is of fixed length 5 with Tag as "ABC", then its length would be displayed here as 8 and not as 5.

If the selected field is a sub-field under a section (Sequence / Choice / All) then an additional information **Position in Section** would be displayed as shown in the following picture.

General	Position	Field Type	Tag	Text Encoding
<b>Field Position</b>				
Position in Message	12			
Position in Data	12			
Position in section	6			
Length	8			

If the selected field is in the Header section, then instead of **Position in Data**, it would read as **Position in Header** as shown in the following picture.



General	Position	Field Type	Tag	Text Encoding
Field Position				
Position in Message		1		
Position in Header		1		
Length		5		

Similarly, if the selected field is in the Trailer section User Interface, then instead of Position in Data, it would read as **Position in Trailer** as shown in the following picture.

General	Position	Field Type	Tag	Text Encoding
Field Position				
Position in Message		18		
Position in Trailer		1		
Length		5		

#### Field Type tab:

The plug-in supports three field types viz., Fixed Length Field, Length Preceded Field and Delimited Field. Under the **Field Type** tab, you can choose the required type of the field by checking the radio button against the desired field type.

#### See Also:

[Adding a Fixed Length field](#)

[Adding a Length Preceded field](#)

[Adding a Delimited field](#)

### Adding a Fixed Length field

If the user wants the length of a particular field to be always of a pre-specified value (fixed), he can do so by checking the **Fixed Length** radio-button under the **Field Type** tab with the required field selected in the table under the External Format User Interface. If the field type for a field is selected as Fixed Length, the required length of that field should be given in the **Length** textbox displayed.

General	Position	Field Type	Tag	Text Encoding
<input checked="" type="radio"/> Fixed Length <input type="radio"/> Length Preceded <input type="radio"/> Delimited				
Length <input type="text" value="5"/>		Length Field <input type="text" value=""/> ▼		Delimiter <input type="text" value="Default"/>
Min Length <input type="text"/>		Quote <input type="text" value="Default"/>		
Max Length <input type="text"/>				

The following points are worth noting while specifying the length for a selected field:

Length should be a positive integer. (Note that the Length textbox is designed to reject non-numeric values.)

It is understood that length of character field should compulsorily be given as 1, otherwise, an error is generated during validation.

In case of a Date field, the specified length should be greater than or equal to the length determined by the selected Format, i.e, if you have selected the Format for the Date field as say, yyyyMMdd (length = 8) then you should specify the length of the field to be  $\geq 8$  only. Otherwise, you would get error on parsing. If the length is specified to be  $>8$ , then, padding has to be given.

In case of a Boolean field, the length is determined by the True and False Values given under the Text Encoding tab. If True and False values are not given, then the values given at Format Options are taken by default. If True and False values are given, then Length is determined by the maximum of those values. For e.g., if you have given True Value as TRUE and False Value as FALSE (maximum length being 5) you have to compulsorily specify the Length to be 5, otherwise, you would get error on validation.

In case of a Text Encoded field of Float, Double or BigDecimal data type, the value if any (specifying format for a Text Encoded field is not mandatory) given in Format textbox under the 'Text Encoding' tab will determine the length of the field. For e.g., if for field of type Float, you have given the value of Format under the Text Encoding tab as 4.3 (meaning 4 digits before decimal and 3 digits after decimal), then you should specify the Length to be 8 (including the decimal point which represents one byte).

For a Fixed Length Binary Encoded field of Integer, Long, Float or Double data type, the following convention should be followed while specifying the length, otherwise, you would get error on validation:

Data Type	Valid Length
Integer	1, 2 or 4
Long	1, 2, 4 or 8
Float	4
Double	8

For a Fixed Length Packed Decimal Encoded field of Integer, Long, Float or Double data type, the Format/Digit specified under the Packed Decimal Encoding tab would determine the length of the field as described in the following table:

Data Type	Valid Format/Digit	Valid Length
Integer	$0 \leq N \leq 10$ (N - # of digits)	$(N+1)/2$ (rounded off)
Long	$0 \leq N \leq 19$	$(N+1)/2$ (rounded off)
Float	M.N (M - # of digits before decimal, N - # of digits after decimal)	$(M+N+1)/2$ (rounded off)
Double	M.N (M - # of digits before decimal, N - # of digits after decimal)	$(M+N+1)/2$ (rounded off)

(The Designer will validate if the specified length is valid as per the above specification and if not, would throw error on validation)

For e.g., for a Fixed Length Packed Decimal field of type integer, if you have given the value for Digits as, say, 8 in the Digits/Format textbox, then the Length should be given as  $(8+1)/2 = 4.5$ , rounded off = 5. In case of a field of type Double, if you have given the value for Format as, say, 7.3 (meaning 7 digits before decimal and 3 digits after decimal) then the length should be given as  $(7+3+1)/2 = 5.5$ , rounded off = 6. (Note that in case of Packed Decimal Encoding, the decimal point is not stored and hence need not be taken into account while specifying the length unlike in the case of a Text Encoded field)

**Note:**

To make a Fixed Length field optional, it is not enough that you uncheck the **Required** check box under the **General** tab, because, you still have to give empty field values (spaces). To make it truly optional, you can add a tag for that field, provided the start value of the subsequent field is not the same as the tag value.

**See Also:**

- [Adding a Length Preceded field](#)
- [Adding a Delimited field](#)

## Adding a Length Preceded Field

A **Length Preceded** field is a variable length field. If the user wants the length of a field to be determined during run-time based on the value of another field of integer type (set as count field) in the message that can be done by checking the **Length Preceded** radio-button under the **Field Type** tab. Note that for a **Length Preceded** field, only **Text Encoding** is supported and Filler fields cannot be of length preceded type.

General	Position	Field Type	Tag	Text Encoding
<input type="radio"/> Fixed Length				
<input checked="" type="radio"/> Length Preceded				
<input type="radio"/> Delimited				
Length <input type="text" value="0"/>		Length Field <input type="text" value="Size"/>		Delimiter <input type="text" value="Default"/>
		Min Length <input type="text"/>		Quote <input type="text" value="Default"/>
		Max Length <input type="text"/>		

### Length Field

The required count field which is to determine the length of the selected field during run-time is to be selected in the Length Field combo-box. At runtime this field specifies the length/size of the Length Preceded Field it applies to. The following points need to be noted while specifying the count field.

1. The count field should be of integer type.
2. It should precede the field for which it is set as a count.
3. The length field and the 'Length Preceded' field must be siblings, i.e., they should have same parent. For e.g., for a sub-field added under a section, you cannot mention as length field, a field that is appearing above that section. However, the above constraint can be overcome by using a formula (use of formula feature is described below) along with the length field. For e.g., for a 'Length Preceded' sub-field added under a sequence A, you can specify the formula  $SIZE + 1$  in the 'Length Field' combo-box, where SIZE is a field appearing above its parent sequence A and thus not a sibling.

**Note:**

When the message is outbound, the Length Field is automatically set to the length of the Length Preceded field it applies to (provided you have simply selected an integer field as the Length Field without using any formula) i.e. you need not provide a mapping for this field. For e.g., in a Universal output message format, let F1 be a 'Length Preceded' string field with length field SIZE of fixed length 5. In output mapping, only F1 needs to be mapped. During runtime, if we set the value of F1 as "AB", then the value of the SIZE field is automatically generated as 00002 (padding set as zero). However, if you have used a formula in the 'Length Field' combo-box, it becomes your responsibility during mapping or some other phase to ensure that the value of the length field and the length of the 'Length Preceded' field are consistent.

**Min Length**

This optional attribute specifies minimum length of the Length Preceded field. The actual length (which is greater than or equal to min length) will be determined by the Length Field.

**Max Length**

This optional attribute specifies maximum length of the Length Preceded field. The actual length (which is less than or equal to max length) will be determined by the Length Field.

**Note:**

If Min Length and/or Max Length properties are set, then the value given for the count field during run-time must be in conformity with the minimum & maximum length set, otherwise, the message would not be parsed. For e.g., let us suppose that Min Length for the Length Preceded field is set as 3 and value of count field given in the incoming message be 2. In this case, the message would not be parsed as the Min Length condition is violated.

## Salient Features of a Length Preceded field

A Length Preceded field is used not only in the context as noted above; it has other interesting possibilities as given below.

### Using Formula feature (Access preceding field)

Though the combo-box is named 'Length Field', you need not actually specify a field there. Note that when the cursor is in this combo-box, the Edit Formula icon is enabled in the toolbar and Edit Formula hyperlink is displayed in the Status Bar, meaning that you can specify a formula in that combo-box. The formula should return an integer value representing the length of the field.

For e.g., if you want the length of the field to be 2 times the value of the SIZE field, then you can specify the same in the 'Length Field' combo-box using the formula **SIZE \* 2**.

The screenshot shows the 'Field Type' tab of a configuration window. It contains three radio button options: 'Fixed Length', 'Length Preceded' (which is selected), and 'Delimited'. Under 'Fixed Length', there is a 'Length' text box containing the value '0'. Under 'Length Preceded', there is a 'Length Field' dropdown menu containing the formula 'SIZE \* 2', and two empty text boxes for 'Min Length' and 'Max Length'. Under 'Delimited', there are two text boxes for 'Delimiter' and 'Quote', both containing the value 'Default'.

### Length Preceded field as a Fixed Length field

By specifying the required length for the field in the Length Field combo-box, you can make the Length Preceded field to act as a simple Fixed Length field. For e.g., if the field is to be of fixed length, say 5, just type 5 in the Length Field combo-box. (A better way of doing this is to make it a Fixed Length field)

This screenshot is identical to the previous one, but the 'Length Field' dropdown menu now contains the value '5' instead of the formula 'SIZE \* 2'. All other elements, including the selected 'Length Preceded' radio button and the 'Default' values in the 'Delimiter' and 'Quote' boxes, remain the same.

### **Length Preceded field as a Delimited field (Peek the rest of the message)**

A Length Preceded field can be made to act as a Delimited field also. For e.g., suppose you want to delimit the value of the field marked as Length Preceded type by a delimiter, say, comma, then you can specify the same in the Length Field combo-box using the formula:

```
FindFirst (PeekRest(), ",")
```

PeekRest() function returns a string, which contains the rest of the characters in the current record. (Refer Universal Plug-in Peek Formula Functions). FindFirst function returns the index of first occurrence of comma in the string returned by PeekRest(). Suppose the string returned is "ABCDE,FGHI", then FindFirst function will return the index as 5 (indexing starts from zero), thus, the length is set as 5 and the next five characters from the current position, i.e., "ABCDE" will be taken as the value of that field. The problem that would arise while using the above formula is, though you have delimited the field by the specified delimiter, the delimiter is not ignored, but is taken as the start value of the subsequent field. To overcome this problem, you have to add a Fixed Value Filler with value as that of the delimiter.

### **To parse Swift fields (Variable Length & Optional) using Length Preceded field of Universal**

In a Length Preceded Field, we can use formula to peek at the input to decide on the length of the field, thus supporting Variable Length fields.

We can also use a formula that would return 0 to indicate that the field can be missing, thus allowing us to support truly Optional fields. (Note that if a FIXED length field is marked as optional, it just means that the field can have null value; the input should still contain required number of characters (as spaces). The truly optional fields that we are talking about will simply be missing in input.)

The above two types of fields are very common in Swift input.

Example

```
(code)(ref) 3n[4!a]
```

Here code is a variable length field, which can have maximum of three digits, while ref is an optional field (as it is specified in square brackets).

Consider the formula function `MatchFormat(value, format)` that returns the number of characters that matched (0 if there is no match)  
For e.g., `MatchFormat("78AAAA..", "3n")` returns 2.

Using this formula in the context of a length preceded field, you can make it parse swift fields as noted below.

```
MatchFormat(PeekRest(), "3n")
```

returns the number of numeric characters (max of 3) that immediately follow the input.

Using this, we can effectively discriminate fields based on the swift char set. Obviously this is not limited to parsing swift input. We can use any other pattern language (such as Regex) to find the presence of a field. Since formula functions are (in theory) extensible we can use this as an escape mechanism to parse complex inputs.

Thus, we can call a Length Preceded field as a superset of Fixed Length and Delimited fields, as the other two field types turn out to be special cases that can be represented as Length Preceded formula. As using the Formula feature in Length Preceded Field, the parser can dynamically peek into the message, most cases, which cannot be addressed by either, delimited or fixed length fields can be implemented using Length Preceded field.

**See Also:**

[Adding a Fixed Length field](#)

[Adding a Delimited field](#)

## Adding a Delimited field

As the name itself indicates, if you want the value of the field to end at a specific point, say, where a comma is encountered (ignoring that comma), then you can do so by checking the **Delimited** radio-button under the **Field Type** tab. Note that for a **Delimited** field, only 'Text Encoding' is supported.



General	Position	Field Type	Tag	Text Encoding
<input type="radio"/> Fixed Length Length <input type="text" value="0"/>				
<input type="radio"/> Length Priceded Length Field <input type="text" value=""/>				
<input checked="" type="radio"/> Delimited Delimiter <input type="text" value=","/>				
<input type="text" value=""/>				
Min Length <input type="text" value=""/>				
Max Length <input type="text" value=""/>				
Quote <input type="text" value=""/>				

## Delimiter

The required Field Delimiter is to be specified in this combo-box. If you have already specified the Field Delimiter using the Format Options feature, then you would find the word 'Default' automatically selected in this combo-box, which can be changed at field level. A delimiter can be of any length and can be a combination of both alphanumeric and special character. If a delimiter is set at field level, that is used, if not, the default set at the section level is used, if not, one set at the format level is used.

## Quote

The Quote combo box provided under Default Field Delimiter is used to specify the escape sequence to be used in case the delimiter character/s is/are to be included in the value of the field. For e.g. let us consider a delimited integer field (non-null) with delimiter specified as 1. Suppose you want to give value for the field as 12345. If you don't give escape sequence, the field would be considered as having NULL value, as the delimiter (1) is encountered in the first position itself. You can overcome this difficulty by giving an escape sequence, say " under the Quote combo box. Now the value of the field should be given as "12345"1 meaning that the 1 encountered in the data should be taken as its actual value and not as the delimiter. Note that the Quote given should be a single character (alphabet, numeric or special character) and should not be the same as the delimiter.

## Tag tab:

Tag is a prefix that is added to the field or section in messages for easy identification (during parsing). The field value starts after the tag ends. A separator can optionally separate the tag and the field value. Note that in case of sections, the tag is a prefix for every instance of the section. Tagged fields are often used in the context of 'All' type sections (unordered sections), where presence of the field must be ascertained before parsing it.

General	Position	Field Type	Tag	Text Encoding
Tag	<input type="text" value="1"/>			
Separator	<input "="" type="text" value="="/>			

## Tag

You can specify the required tag in the **Tag** textbox. A tag can be of any length, can contain both alphanumeric and special character. Remember that a tag, in most cases, should be unique when specified for fields at the same level. If not, the message may not be parsed correctly (though the Designer does not check for uniqueness of tag during validation). If Tag is specified for a field in Input Message Format, while giving value for the field, Tag should compulsorily precede the value. In case of Output Message Format, Tag will be automatically written as a prefix of the value.

## Separator

You can specify the required separator in the **Separator** textbox. This property is optional.

For e.g., if you want a field A to be identified by the tag, say 1 and the tag and the field value to be separated by a separator, say = then you should specify 1 in the Tag textbox and = in Separator textbox. In the message, if a field is encountered with value "1=ABCD" then the value "ABCD" would be assigned to the field A. Note that for a delimited field, the tag or the separator should not be the same as the specified delimiter.

## Encoding tab

The name of this tab dynamically changes to Text Encoding, Binary Encoding or Packed Decimal Encoding as per the Encoding option set for the selected field. All the three encoding types (Text, Binary and Packed Decimal) are applicable for fields of Integer, Long, Float and Double data type. For Binary and Boolean fields, both Binary Encoding and Text Encoding are applicable. For all other data types, only Text Encoding is applicable.

## See Also:

[Adding a Fixed Length field](#)

[Adding a Length Preceded field](#)

## Encoding Types

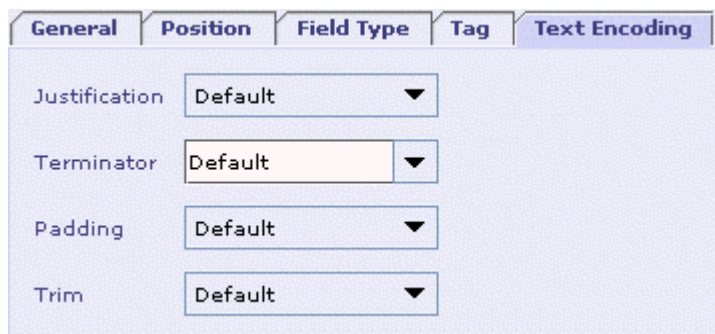
Three types of encoding are supported in Universal, [Text Encoding](#), [Binary Encoding](#) and [Packed Decimal Encoding](#). A field should have one of the three types of encoding.

### Text Encoding

The various properties that can be set for a Text Encoded field dynamically changes in the properties pane depending on the data type of the selected field.

#### Text Encoded String field

If the selected Text Encoded field is of String data type, the properties that can be specified under the 'Text Encoding' tab are as shown in the following picture.

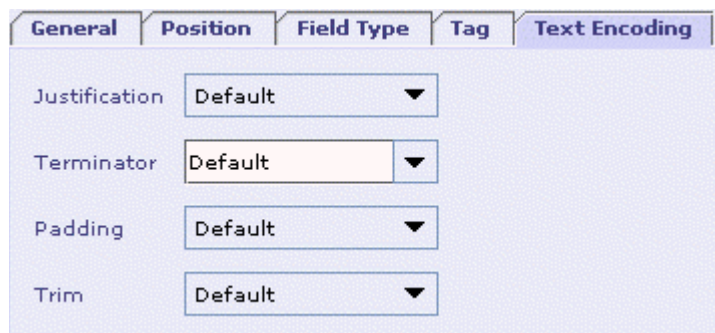


The screenshot shows a software interface with five tabs: 'General', 'Position', 'Field Type', 'Tag', and 'Text Encoding'. The 'Text Encoding' tab is selected and active. It contains four rows of settings, each with a label on the left and a dropdown menu on the right. All dropdown menus are currently set to 'Default'. The settings are: Justification (Default), Terminator (Default), Padding (Default), and Trim (Default).

These properties have already been elaborated upon in the [Format Options](#) section.

#### Text Encoded Character field

If the selected Text Encoded field is of Character data type, the properties that can be specified under the 'Text Encoding' tab are as shown in the following picture.

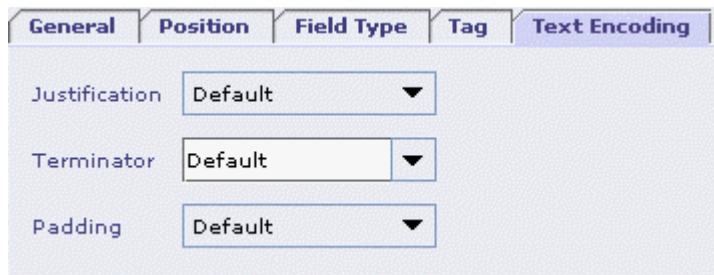


The screenshot shows a software interface with five tabs: 'General', 'Position', 'Field Type', 'Tag', and 'Text Encoding'. The 'Text Encoding' tab is selected and active. It contains four rows of settings, each with a label on the left and a dropdown menu on the right. All dropdown menus are currently set to 'Default'. The settings are: Justification (Default), Terminator (Default), Padding (Default), and Trim (Default).

These properties are the same as for a String field. These properties have already been elaborated upon in the [Format Options](#) section.

### Text Encoded Integer field

If the selected Text Encoded field is of Integer data type, the properties that can be specified under the 'Text Encoding' tab are as shown in the following picture.



General	Position	Field Type	Tag	Text Encoding
Justification	Default			
Terminator	Default			
Padding	Default			

Note that except for the Trim property, all other properties are same as that for a string field, however for a numeric data type, two padding options namely Zeros (applies only if field is right justified) and Space are available. To learn about the connection between Justification and Padding properties for an integer field, see [Note](#) under Format Options section (Justification).

### Text Encoded BigInteger field

The properties that can be specified for a Text Encoded BigInteger field are same as that of an Integer field.

### Text Encoded Long field

The properties that can be specified for a Text Encoded Long field are same as that of an Integer field.

### Text Encoded Float field

If the selected Text Encoded field is of Float data type, the properties that can be specified under the 'Text Encoding' tab are as shown in the following picture.

### Decimal Point

Decimal Point property has already been dealt with in Format Options section. See [Decimal Point](#) in that section.

### Format

You can specify the required format of the field in the Format textbox as M.N where M represents # of digits before decimal and N represents # of digits after decimal. For e.g. if Format is specified as 3.2, it would mean that there should be 3 digits before the decimal point and 2 digits after. Note that the dot specified in the format M.N has nothing to do with the Decimal Point selected, i.e., even if you have chosen the Decimal Point to be Comma (,) in the Format you should give as M.N only. Specifying the Format is optional for a Text Encoded float field unlike a Packed Decimal Encoded field, however, if the Decimal Point option is selected as None, then the Format should compulsorily be given, because, the real and fractional parts of the field could not be distinguished otherwise.

For a Fixed Length field, if Format is specified, then Length should match the Format and if Decimal Point is selected as None, then you need not include the decimal point in the length. The following table would clearly explain this:

Format	Decimal Point	Length
0.1	Dot	2
0.1	None	1
1.0	Dot	2
1.0	None	1
5.3	Dot	9

4.2	Comma	7
3.3	None	6 (Not 7)

For a Delimited field, the Delimiter and the Decimal Point should not be the same. If given so, then Delimiter overrides the Decimal Point and the data would not be parsed.

### Exp Rep

If you want to allow exponential representation for the float field, you can check this check box. Note that this property should not be set to true if Decimal Point is selected as None. For e.g., if you have set this property to be true, you can give the value for a float field as 1.5e4 to represent 15000.0

### Text Encoded Double field

The properties that can be specified for a Text Encoded Double field are same as that of a Float field.

### Text Encoded BigDecimal field

The properties that can be specified for a Text Encoded BigDecimal field are same as that of a Float field.

### Text Encoded DateOnly field

If the selected Text Encoded field is of DateOnly data type, the properties that can be specified under the Text Encoding tab are as shown in the following picture.

The screenshot shows a configuration window with five tabs: 'General', 'Position', 'Field Type', 'Tag', and 'Text Encoding'. The 'Text Encoding' tab is active. It contains two dropdown menus. The first is labeled 'Justification' and is set to 'Default'. The second is labeled 'Terminator' and is also set to 'Default'.

These properties viz., Justification and Terminator have already been dealt in detail in [Format Options](#) section.

### Text Encoded DateTime field

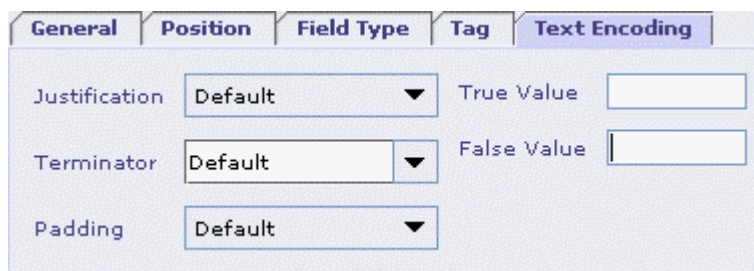
The properties that can be specified for a Text Encoded DateTime field are same as that of a DateOnly field.

### Text Encoded TimeOnly field

The properties that can be specified for a Text Encoded TimeOnly field are same as that of a DateOnly field.

### Text Encoded Boolean field

If the selected Text Encoded field is of Boolean data type, the properties that can be specified under the 'Text Encoding' tab are as shown in the following picture.



General	Position	Field Type	Tag	Text Encoding
Justification	Default		True Value	<input type="text"/>
Terminator	Default		False Value	<input type="text"/>
Padding	Default			

The properties viz., Justification and Terminator have already been dealt with in detail in [Format Options](#) section.

The 'True Value' and 'False Value' text-boxes are used to specify the true and false values to be set for a Boolean field. This property is mandatory. In case of a Boolean field of Fixed Length type, the Length is determined by the True and False Values given under the Text Encoding tab. If True and False values are not given, then the values given at Format Options are taken by default. If True and False values are given, then Length is determined by the maximum of those values. For e.g., if you have given True Value as TRUE and False Value as FALSE (maximum length being 5) you have to compulsorily specify the Length to be 5, otherwise, the Designer would throw error message on validation.

### Text Encoded Binary field

If the selected Text Encoded field is of Binary data type, the properties that can be specified under the 'Text Encoding' tab are as shown in the following picture.

General	Position	Field Type	Tag	Text Encoding
Justification	Default			
Terminator	Default			
Formats	Default			

The properties viz., Justification and Terminator have already been dealt in detail in [Format Options](#) section.

### Formats

A Text encoded Binary field can be specified one of two formats – 'Base64' or 'Hex'. The field value should correspond to the specified format.

## Binary Encoding

This Encoding is used to store quantities in binary format. Binary uses either Word (4 bytes), Half-word (2 bytes) or Double-word (8 bytes), based on precision. Half-word can store  $2^{15}$  to  $(2^{15}-1)$  or  $2^{16}$ , depending upon whether the number is signed or unsigned.

**Field Type supported** - Fixed Length and Length Preceded.

**Data Types supported** - Integer, Long, Float, Double, Binary and Boolean.

### Note:

The Designer does not allow a binary encoded field to be set as optional, i.e., for a binary encoded field, you should not uncheck the Required check-box under the General tab.

A Section cannot have delimiter if a field in the section is binary encoded.

### Binary Encoded Integer or Long field

If the selected Binary Encoded field is of data type Integer or Long, the properties that can be specified under the **Binary Encoding** tab are as shown in the following picture.





### Endian

The **Endian** listbox is used to specify the Endian type (which represents the byte order specific to the Microprocessor) to be used for the Binary Encoded field. The user can select between the two options available namely Little Endian and Big Endian.

### Unsigned

'Unsigned' indicates whether the input/output should be treated as unsigned. This is applicable only for Integer and Long fields and not applicable for Float and Double fields.

Note that the following restriction should be applied while specifying the Length under the Field Type tab, otherwise, the Designer would throw error on validation:

Data Type	Valid Length
Integer	1, 2 or 4
Long	1, 2, 4 or 8

The following limitation is present while specifying unsigned types. Integer fields cannot be specified as 'unsigned'. If you specify an integer field as unsigned it will result in validation error.

This is because the integer type in Designer is signed integer. Designer, like Java, does not have an 'unsigned int' type. If you have specified the type as integer, unsigned with four bytes, the upper half of the range you have chosen will overflow if it is represented as 'Integer'. Solutions to this limitation would be

- Using Long with unsigned option.
- Using Integer if you are sure about the range.

### Binary Encoded Float or Double field

If the selected Binary Encoded field is of data type Float or Double, the only property that can be specified under the **Binary Encoding** tab is **Endian** (explained above) as shown in the following picture.



**Float** - IEEE 32 bit representation

**Double** - IEEE 64 bit representation

Note that the following restriction should be applied while specifying the length, otherwise, the Designer would throw error on validation:

Data Type	Valid Length
Float	4
Double	8

### Binary Encoded Binary field

For a 'Binary Encoded' field of Binary data type, no encoding options can be specified. The field value is represented as raw bytes.

### Binary Encoded Boolean field

If the selected Binary Encoded field is of Boolean data type, the only property that can be specified under the **Binary Encoding** tab is **Endian**. The user can select between the two options available namely 'Little Endian' and 'Big Endian'.

Please note that a Binary Encoded field of Boolean data type can be used only with ['Fixed Length' field type](#) and its length should be 1. The byte value of 0 (zero) represents **false** and anything other than zero represents **true**.

### Packed Decimal Encoding

In this type of Encoding, each digit 0 through 9 is represented in 4 bits. Hence 2 digits are packed into each byte. Rightmost 4 bits of the number store sign: "F" for unsigned, "D" for minus and "C" for plus. Packed-decimal numbers may have up to 18 digits. The byte length is equal to  $(N+1)/2$  rounded up, where N is the number of digits in the value with one added for sign.

E.g.:

564.22 is stored as |56|42|2F| (each slot represents 1 byte). (-564.22) is stored as |56|42|2D|. **Note that the decimal point is not stored.**

**Field Type supported** - Fixed Length

**Data Types supported** - Integer, Long, Float and Double

For a Packed Decimal Encoded field, the property that has to be specified under the **Packed Decimal Encoding** tab is as shown in the following picture:

The image shows a software interface with five tabs: 'General', 'Position', 'Field Type', 'Tag', and 'Packed Decimal Encoding'. The 'Packed Decimal Encoding' tab is active. Below the tabs, there is a label 'Digits/Format' followed by a rectangular text input box.

### Digits/Format

In this textbox, the Digits (applicable for Integer and Long data types) or the Format (applicable for Float and Double data types) has to be specified. This is a mandatory property and would determine the length of the field.

### Note:

The Designer does not allow a packed decimal encoded field to be set as optional, i.e., for a packed decimal encoded field, you should not uncheck the Required checkbox under the General tab.

### Packed Decimal Encoded Integer or Long field

For a field of this type, you should compulsorily specify the required number of **Digits** (N) in the Digits/Format textbox, which would determine the length of the field as per the formula described in the following table:

Data Type (Valid N)	Digits (N)	Valid Length $(N+1)/2$ (rounded off)
Integer ( $0 \leq N \leq 10$ )	0	1
Integer	1	1
Integer	5	3
Integer	10	6
Long ( $0 \leq N \leq 19$ )	0	1
Long	8	5
Long	12	7
Long	18	10
Long	19	10

While specifying the Length under the Field Type tab, the above restriction has to be applied, else, the Designer would throw error message on validation.

### Packed Decimal Encoded Float or Double field

For a field of this type, you should compulsorily specify the required **Format** (M.N) where M denotes # of digits before decimal point and N denotes # of digits after decimal point, in the Digits/Format textbox, which would determine the length of the field as per the formula described in the following table:

Data Type	Digits (M.N)	Valid Length $(M+N+1)/2$ (rounded off)
Float / Double	0.0	1
Float / Double	0.1	1
Float / Double	3.1	3
Float / Double	5.3	5
Float / Double	10.5	8

While specifying the Length under the Field Type tab, the above restriction has to be applied, else, the Designer would throw error message on validation. (Note that in case of Packed Decimal Encoding, the decimal point is not stored and hence the Designer does not take that into account while determining the length of the field unlike the case of a Text Encoded field where the decimal point is taken into account while determining the length except where it is set as None.)

#### See Also:

[Adding Fillers and specifying properties](#)

[Adding a Sequence type section and specifying properties](#)

[Adding a Choice type Section and specifying properties](#)

[Adding an All type Section and specifying properties](#)

[Universal Format Options](#)

## Adding Fillers and specifying properties

A Filler is a special field. It is basically used to skip over an area in the message (which may be used later to accommodate fields without changing the length of the message). It acts as a tool for message evolution.

### Attributes of a Filler field


A Filler field is always of String data type.

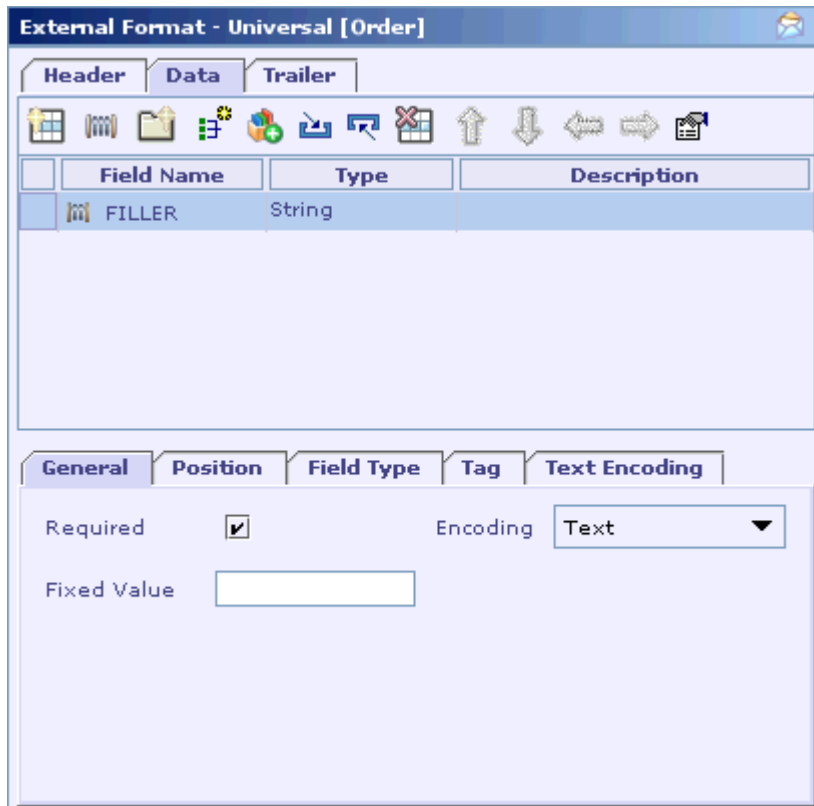
Length Preceded field type not supported for fillers.

A Filler field cannot be mapped and will not appear in mapping and validation screens. It also does not have a runtime representation.

Filler fields are exempt from name uniqueness, i.e., you can have more than one Filler field with the same name.

### To add a Filler in the Universal - External Format UI

1. Click the **Add Filler** button  in the Universal - External Format UI.
2. A new row (Filler) is added in the External Format UI with default name 'FILLER' as shown below.



You can change the field name as per your requirement by double-clicking on the column under "Field Name". The default type namely "String" which is generated cannot be modified because, a FILLER field can be of type String only.

## Specifying properties for a Filler

The properties panel that appears in the bottom pane of the UI when a Filler is selected is identical to that of a normal field (which is explained in detail in 4.2.1) except that under the **General** tab, instead of Default Value textbox, which appears for a field, **Fixed Value** textbox appears as shown in the above picture.

### Fixed Value

This property is optional and can be used if the user wants to specify a fixed value for the Filler instead of just giving spaces (applicable for Input Message) or wants the Filler field to contain a pre-specified Fixed Value instead of spaces (applicable for Output Message)

For a Fixed Length Filler, if a Fixed Value is specified under the General tab, its width should match the specified Length. Otherwise, the Designer would throw validation error. For e.g., let the Length given for a Fixed Value FILLER be 5.

Then if a Fixed Value is specified for that FILLER under the General tab, it should be in the form, say, "ABCDE" (Width = 5).

If a Fixed Value Filler is added in a Universal Input Message format, during parsing, the Designer would validate if the value in the message matches the specified Fixed Value and would skip over that many characters.

If a Fixed Value Filler is added in a Universal Output Message format, during runtime, the Designer automatically assigns the specified Fixed Value to the value of the Filler.

Escape sequence is supported while specifying the Fixed Value. For e.g., you can use a filler for giving Carriage Return & Line Feed by specifying `\r\n` in the Fixed Value textbox (Length to be given as 2). Refer [APPENDIX](#) to find out the various escape sequences that can be used while specifying the Fixed Value.

## Properties applicable for a Filler field

### General Tab

**Required** check box is used to specify if the selected filler is mandatory or not.

**Fixed Value** property is explained above.

In **Encoding** list box, only Text Encoding would be applicable and would be selected by default.

### Position Tab

Same as explained for a normal field. Refer ['Position' tab](#) explained in 4.2.1.

### Field Type Tab

For a **Fixed Length** Filler (similar to that of a [Fixed Length field](#) explained in 4.2.1) the value of **Length** should match the width of Fixed Value, if given.

**Length Preceded** Filler not supported.

For a **Delimited** Filler added in a Universal External Format, the Filler (empty spaces or Fixed Value) should end with specified delimiter.

### Tag Tab

Same as explained for a normal field. Refer ['Tag' tab](#) explained in 4.2.1.



## Text Encoding Tab

The properties appearing under this tab are not applicable for a Filler field.

## Usage of a Filler

A Fixed Value Filler added as the first element of a Section acts as a tag for that section.

A Fixed Value Filler added as the first child element of a section or at a fixed location (preceded by fixed length fields) can be used for detecting the presence of a discriminated section (instead of specifying a formula).

### See Also:

[Adding Fields and specifying properties](#)

[Adding a Sequence type section and specifying properties](#)

[Adding a Choice type Section and specifying properties](#)


[Adding an All type Section and specifying properties](#)

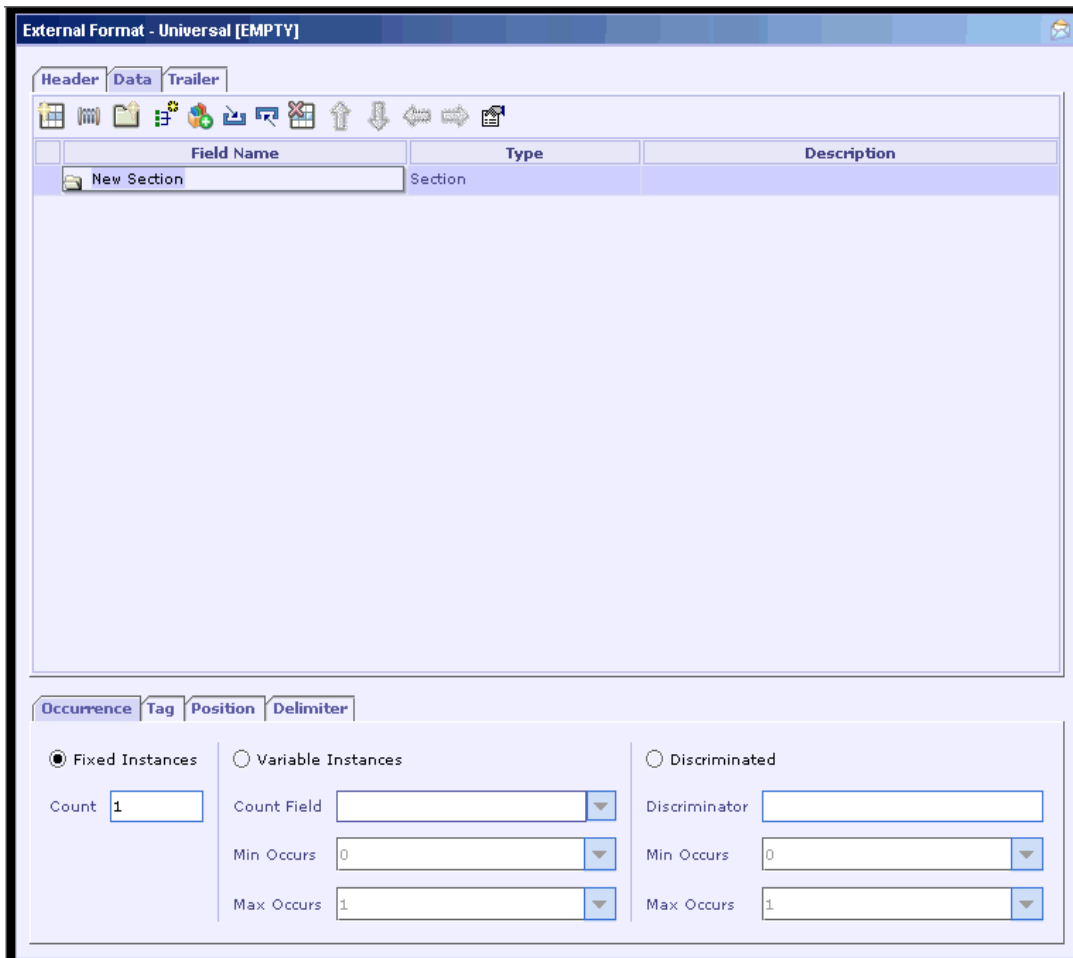
[Universal Format Options](#)

## Adding a *Sequence* type section and specifying properties

A **Sequence** is a group of related information. A Sequence can have as its child elements another Sequence / Field / Filler / Choice / All. As the name itself indicates, in a Sequence, all the child elements should occur in the specified order, i.e. a Sequence can be called as an ordered section.

### To add a Sequence in the Universal - External Format UI

1. Click the **Add Sequence button**  in the Universal - External Format UI.
2. A new row (Sequence) is added in the External Format UI with default name 'New Section' and default type set as 'Section' as shown below.



The screenshot shows the 'External Format - Universal [EMPTY]' interface. At the top, there are tabs for 'Header', 'Data', and 'Trailer'. Below the tabs is a toolbar with various icons. A table with three columns: 'Field Name', 'Type', and 'Description' is visible. The table contains one row with 'New Section' in the 'Field Name' column and 'Section' in the 'Type' column. Below the table, there are four tabs: 'Occurrence', 'Tag', 'Position', and 'Delimiter'. The 'Occurrence' tab is active, showing three radio buttons: 'Fixed Instances' (selected), 'Variable Instances', and 'Discriminated'. Below these are several input fields and dropdown menus for 'Count', 'Count Field', 'Min Occurs', and 'Max Occurs' for both the 'Fixed Instances' and 'Discriminated' options.

Field Name	Type	Description
New Section	Section	

Fixed Instances   
  Variable Instances   
  Discriminated

Count:    
 Count Field:    
 Discriminator:

Min Occurs:    
 Min Occurs:

Max Occurs:    
 Max Occurs:

You can change the Sequence name as per your requirement by double-clicking on the column under "Field Name". Note that there cannot be duplicate sequences (sequences with same name) at the same level.

3. Under the column **Alias** (which is hidden in the above picture), you can specify the alias (substitute name) to be used for the Sequence when specifying a Validation/Mapping formula involving that Sequence. This property is optional. (If a Sequence is assigned an alias, the qualified name of that Sequence is replaced with its newly assigned alias in all the places it is used.)
4. Under the column **Description**, you can mention any description for the added Sequence. This property is also optional.

### Specifying properties for a Sequence

The Sequence properties panel has four tabs,

Property Tab	Usage
Occurrence	Can specify the number of times the sequence occurs in the message & find out the presence of an optional sequence
Tag	Can specify the tag and separator for the sequence
Position	Displays the position of the sequence in the message, data and its length
Delimiter	Can specify the Delimiter for the sequence as well as the Default Field Delimiter (for the fields added under it)

**Note:**

The above properties are equally applicable to any Section, viz., Sequence / Choice / All.

**Occurrence tab:**

This property is somewhat similar to the Field Type tab of a field. Universal plug-in supports creating three types of sequences viz., Fixed Instances sequence, Variable Instances sequence and Discriminated sequence. Under the **Occurrence** tab, you



can choose the required type of the sequence by checking the corresponding radio button. (Note that this property is common for any Section – Sequence/Choice/All)

**See Also:**[Adding a Fixed Instances sequence](#)[Adding a Variable Instances sequence](#)[Adding a Discriminated sequence](#)[Adding a Choice type Section and specifying properties](#)[Adding an All type Section and specifying properties](#)

## Adding a Fixed Instances Sequence

If the user wants a particular sequence in the message to always occur a specific number of times, that can be done by checking the **Fixed Instances** radio-button under the **Occurrence** tab with the required sequence selected in the table under the External Format User Interface. If the Occurrence property for a sequence is selected as Fixed Instances, the required number of occurrences for that sequence should be specified in the **Count** textbox displayed.

The screenshot shows the 'Occurrence' tab of the External Format User Interface. It features three radio buttons: 'Fixed Instances' (selected), 'Variable Instances', and 'Discriminated'. Under 'Fixed Instances', there is a 'Count' text box containing the value '3', and three dropdown menus for 'Count Field', 'Min Occurs' (set to 0), and 'Max Occurs' (set to 1). The 'Discriminated' section includes a 'Discriminator' text box and three dropdown menus for 'Min Occurs' (set to 0) and 'Max Occurs' (set to 1). The 'Tag', 'Position', and 'Delimiter' tabs are visible at the top of the interface.

### Count

In this textbox, you have to enter an integer value specifying the required number of instances for the sequence. On the outbound side, a runtime exception is generated if the instances created do not match the specified number. (This is metadata only – doesn't appear in the message.)

**Note:**

You can create a non-repeating sequence by giving value of count as 1.

You can create a repeating sequence that repeats exactly 5 times, by giving value of count as 5.

**See Also:**[Adding a Sequence type section and specifying properties](#)

## Adding a Variable Instances Sequence

If the user wants the number of occurrences of a sequence to be determined during run-time based on the value of another field of integer type (set as count field) in the message, that can be done by checking the **Variable Instances** radio-button under the **Field Type** tab.

The screenshot shows the configuration interface for adding a variable instances sequence. The 'Field Type' tab is selected, and the 'Variable Instances' radio button is chosen. The 'Count Field' is set to 'SecCount', and the 'Max Occurs' is set to 'Unbounded'. The 'Discriminated' section is also visible, with 'Min Occurs' set to 0 and 'Max Occurs' set to 1.

### Count Field

The required count field which is to determine the number of occurrences of the selected sequence during runtime is to be selected in the **Count Field** combo-box. This has to be a pre-defined field in the message of Integer type. At runtime this field specifies the cardinality of the repeating sequence it applies to.

The following points need to be noted while specifying the count field.

1. The Count Field should be a mandatory field of integer type.
2. It should precede the sequence for which it is set as a count.
3. The Count Field and the Variable Instances section must be siblings, i.e., they should have same parent. For e.g., for a sub-sequence B added under a sequence A, you can't specify as Count Field, a field that is appearing above the parent sequence A. However, the above constraint can be overcome by using a formula (Use of Formula feature is described below) along with the count field. For e.g., for the sub-sequence B, you can specify the formula `COUNT + 1` in the Count Field combo-box (where COUNT is a field appearing above its parent sequence A and thus not a sibling)
4. Depending upon location of the Count Field within the message definition, different behavior is possible as noted below:

### Fixed instances of child section

ParentCount	Integer counter field
ChildCount	Integer counter field
<b>Parent Section</b>	Repeating depending upon ParentCount
<b>Child Section</b>	Repeating depending upon Childcount Provided formula is used as per (iii)

In this case, every instance of Parent Section will have exactly the same number of instances of Child Section.

For e.g., consider the following Universal Input message structure:

Field Name	Type
ParentCount	Integer
ChildCount	Integer
ParentSection	Section
ChildSection	Section
Field	String

Let the specification for the above message be as under:

ParentCount	Fixed Length field of Length 1
ChildCount	Fixed Length field of Length 1
ParentSection	Variable Instances sequence with Count Field as ParentCount
ChildSection	Variable Instances sequence with Count Field as (ChildCount + 1) (Formula is used)





Let the specification for the above message be as under:

ParentCount	Fixed Length field of Length 1
ParentSection	Variable Instances sequence with Count Field as ParentCount
ChildCount	Fixed Length field of Length 1
ChildSection	Variable Instances sequence with Count Field as ParentSection.ChildCount
Field	Fixed Length Field with Length 1

During runtime, let the values of the Count Fields be as under:

ParentCount = 2  
 For 1<sup>st</sup> Instance of ParentCount, ChildCount = 2  
 For 2<sup>nd</sup> Instance of Parentcount, ChildCount = 3

Now, the sequence ParentSection should appear exactly 2 times and for the first instance of ParentSection, the sub-sequence ChildSection should appear exactly 2 times and for the second instance of ParentSection, the sub-sequence ChildCount should appear exactly 3 times. (Variable instances of child section)

(Note that the term Section is used instead of Sequence because, this property is applicable to any section, be it Sequence/Choice/All.)

**Note:**

When the message is **outbound**, the Count Field is automatically set to the cardinality of the Variable Instances sequence it applies to (provided you have simply selected an integer field as the Count Field without using any formula) i.e. you need not provide a mapping for this field. For e.g., in a Universal Output Message format, let S be a Variable Instances sequence with Count Field COUNT of delimited type with delimiter as comma. In Output Mapping, only the sequence S needs to be mapped. During runtime, suppose we get the number of occurrences of S as 3, then the value of COUNT is automatically generated as 3,

However, if you have used a formula in the Count Field combo-box, it becomes your responsibility during mapping or some other phase to ensure that the value of the Count Field and the number of occurrences of the Variable Instances sequence are consistent.

**Min Occurs**

This optional attribute specifies minimum number of occurrences of the Variable Instances sequence. The actual number of occurrences (which is greater than or equal to this value) will be determined by the Count Field.

**Max Occurs**

This optional attribute specifies maximum number of occurrences of the Variable Instances sequence. The actual number of occurrences (which is less than or equal to this value) will be determined by the Count Field.

**Note:**

If Min Occurs and/or Max Occurs properties are set for a sequence, then the value given for the count field during runtime must be in conformity with the minimum & maximum occurrences set, otherwise, the message would not be parsed. For e.g., let us suppose that Min Occurs property for the Variable Instances sequence is set as 3 and value of count field given in the incoming message be 2. In this case, the message would not be parsed as the Min Occurs condition is violated.

**Salient Features of a Variable Instances sequence**

A Variable Instances sequence is used not only in the context as noted above; it has other interesting possibilities as given below.

**Using Formula feature**

Though the combo-box is named **Count Field**, you need not actually specify just a field there. Note that when the cursor is in this combo-box, the Edit Formula icon is enabled in the toolbar and Edit Formula hyperlink is displayed in the Status Bar, meaning thereby that you can specify a formula here. The Formula should return an integer value representing the number of occurrences of the sequence. Since the formula can peek at the input to decide on the number of occurrences, variable instances sequences are effectively supported.

For e.g., if you want the number of occurrences of a sequence to be the sum of three preceding integer fields, say C1, C2 & C3, then you can specify the same in the Count Field combo-box using the formula: **C1+C2+C3**.

### Variable Instances Sequence as a Fixed Instances Sequence

By specifying the required number of occurrences for the field in the Count Field combo-box, you can make the Variable Instances Sequence to act as a simple Fixed Instances Sequence, provided the values given in Min Occurs and Max Occurs combo-boxes are consistent. For e.g., if a Sequence is to occur a fixed number of times, say 5, just type 5 in the Count Field combo-box.

### To add an Optional Sequence using Variable Instances property

We can also use a formula that would return 0 to indicate that the sequence can be missing, thus allowing us to add Optional sequences.

In the above formula, we have set the sequence to be optional based on the value of an integer field A.

**See Also:**

[Adding a Sequence type section and specifying properties](#)

## Adding a Discriminated sequence

Using this property, you can add optional sequences and repeating sequences whose count is not known. Discriminated sequences (sections) are sections, which are optional and/or repeating, but the number of occurrences is not known at design time, nor is it specified in the message. The presence of the section has to be determined by looking at the input (peeking the message etc.).

Occurrence	Tag	Position	Delimiter
<input type="radio"/> Fixed Instances Count: <input type="text"/>	<input type="radio"/> Variable Instances Count Field: <input type="text" value="F1"/>	<input checked="" type="radio"/> Discriminated Discriminator: <input type="text" value='Peek(1) == "M"    Peek(1) == "F"'/>	
	Min Occurs: <input type="text" value="0"/>	Min Occurs: <input type="text" value="0"/>	
	Max Occurs: <input type="text" value="2"/>	Max Occurs: <input type="text" value="Unbounded"/>	

### Discriminator

You have to specify the required Discriminator in this textbox. The Discriminator specified should be a Boolean formula (should return true or false value), which can make use of all the preceding fields and can also peek at the rest of the record/message and return true if it detects the presence of a sequence (section) instance in the input. (Note that when the cursor is in this textbox, the Edit Formula icon is enabled in the toolbar and Edit Formula hyperlink is displayed in the Status Bar)

For e.g., consider an untagged sequence without any Fixed Value Filler added to it. Suppose the first field in the sequence is of Fixed Length 1 that can have only "M" or "F" as its value, then the occurrence of the sequence can be found out using the following formula as the Discriminator:

**Peek(1) == "M" || Peek(1) == "F"**

**Peek(1)** returns 1 character from the current parser location. (Refer [Universal Plug-in Peek Formula Functions](#)).

While specifying such a Discriminator formula, the user has to make sure that the Discriminator does not result in any other conflicts (For e.g. the message structure having a subsequent field with tag "M" or "F"). The Designer or run-time does not check these conflicts. It is the responsibility of the user or the message Designer to ensure that these conflicts are taken care of.

### Min Occurs

This optional attribute specifies minimum number of occurrences of the Discriminated sequence. The actual number of occurrences (which is greater than or equal to this value) will be determined by the Discriminator.

### Max Occurs

This optional attribute specifies maximum number of occurrences of the Discriminated sequence. The actual number of occurrences (which is less than or equal to this value) will be determined by the Discriminator.

### Note:

If Min Occurs and/or Max Occurs properties are set for a sequence, then the number of occurrences of the sequence as detected by the Discriminator during runtime must be in conformity with the minimum & maximum occurrences set, otherwise, the message would not be parsed. For e.g., let us suppose that Min Occurs property for the Discriminated sequence is set as 3 and the specified Discriminator detects only 2 occurrences of the sequence in the incoming message. In this case, the message would not be parsed, as the Min Occurs condition is violated.

The following points are worth noting while adding a Discriminated Sequence:

If the Discriminated sequence is tagged, then you need not specify a Discriminator formula as the tag acts as an explicit discriminator.

If the Discriminated sequence has a Fixed Value Filler added as the first child element or at a fixed location (preceded by fixed length fields) then you need not specify a Discriminator formula.

### Tag tab:

Tag is a prefix that is added to the sequence (section) in messages for easy identification (during parsing). The sequence (section) starts after the tag ends. A separator can optionally separate the tag and the sequence. If a tag is specified, then it acts as a prefix for every instance of the sequence (section) in case of a repeating sequence. In case of a Discriminated sequence (section) a tag can act as an explicit discriminator and you need not enter a discriminator formula. Tag would be of great help while adding an All type section (unordered section).

Occurrence	Tag	Position	Delimiter
Tag	<input type="text" value="SEC"/>		
Separator	<input type="text" value=":"/>		

## Tag

You can specify the required tag in the **Tag** textbox. A tag can be of any length, can contain both alphanumeric and special character. Remember that a tag name should be unique when specified for sections at the same level. If not, the message would not be parsed. If Tag is specified for a section in Input Message Format, while giving data, you should prefix each occurrence of the section with the specified tag. In case of Output Message Format, Tag will be automatically written as a prefix for every occurrence of the section.

## Separator

You can specify the required separator in the **Separator** textbox. This property is optional.

For e.g., if you want a Section S to be identified by the tag, say SEC and the tag and the Section value to be separated by a separator, say colon (:) then you should specify SEC in the Tag textbox and : in Separator textbox. In the message, if a Section were encountered starting with "SEC:" then the parser would recognize it as section S. Note that for a delimited section, the tag or the separator should not be the same as the specified delimiter.

## Position tab:

Under this tab, the position of the selected sequence (section) and its length are displayed. The position counting starts from zero. The values displayed under this tab are read-only and they change dynamically according to the position of the selected section. This property is generally of use if the message entered is of ASCII-Fixed format. Under this tab, the following properties are displayed for a selected section:

**Position in Message** which is determined by the position of the selected section in the Message, i.e. including Header.

**Position in Data** which is determined by the position of the selected section in the Data. Note that if there is no Header at all for the message format, then this value and the value against **Position in Message** would be one and the same.

**Length** which is determined by the length of the section, which in turn is determined by the length of its child elements. This value is meaningful only if all the child elements of the section are of fixed length.

Occurrence	Tag	Position	Delimiter
Field Position			
	Position in Message	5	
	Position in Data	5	
	Length	2	

If the selected section is a sub-section under a section (Sequence / Choice / All) then an additional property **Position in Section** would be displayed as shown in the following picture.

Occurrence	Tag	Position	Delimiter
Field Position			
	Position in Message	6	
	Position in Data	6	
	Position in section	1	
	Length	2	

If the selected section is in the Header section User Interface, then instead of Position in Data, it would read as **Position in Header** as shown in the following picture.

Occurrence	Tag	Position	Delimiter
Field Position			
	Position in Message	0	
	Position in Header	0	
	Length	1	

Similarly, if the selected section is in the Trailer section User Interface, then instead of Position in Data, it would read as **Position in Trailer** as shown in the following picture.

Field Position	
Position in Message	2
Position in Trailer	0
Length	2

**Delimiter tab:**

Under this tab, you can specify the Delimiter to be applied for the selected section as well as the Default Field Delimiter and Quote, if any, for the sub-fields added under that section.

Delimiter: ,  Ignore Trailing Fields  
 Default Field Delimiter:  
 Delimiter: ;  
 Quote: "

**Delimiter (Section)**

The Delimiter combo box displayed under the **Delimiter** tab is used to specify how the selected Section (Sequence / Choice / All) is to be separated from the subsequent field / section and how different instances of the section have to be separated from one another in case of repeating sections. Sections may be delimited by "\n" or "\r\n" or any other specified literal. Special characters are represented as "\#xH", where H represents a hex number which is a Unicode number. For example, the form feed character (Unicode #xC) can be represented by "\#xC". The line feed character (Unicode #xA) can be represented as "\#xA". This property is optional.

**Note:**

A delimiter can be of any length and can be a combination of both alphanumeric and special character.

If a Delimiter is set at Section level, it is used. If not, one set at Format Level is used (Provided you have selected "Default" from the Delimiter combo-box under the Delimiter tab).



If different Delimiters are set at Section level and Format level, the Delimiter set at Section level overrides the one set at Format level.

For e.g., let the Delimiter specified at Section level be Comma (,) and the one specified at Format Level be semi-colon (;). While giving data, only if you use Comma to delimit the section, the message would be parsed and the message would not be parsed if you use semi-colon to delimit the section.

If the Header/Data/Trailer ends with a section added at the root level (not a sub-section), then it is delimited by the Header/Data/Trailer delimiter if any, specified at the Format Level, i.e., in the Input Message, you need not give the section delimiter for the last section.

If a section has a delimiter, then the value of the fields added under it should not contain the delimiter's character sequence. For e.g., if a section is delimited by `\r\n`, then the fields under it cannot have this combination of characters.

## Ignore Trailing Fields

This checkbox can be checked if the optional trailing fields of a **delimited** Section are to be ignored. This property is used to avoid giving empty field values (spaces) in case the optional trailing fields are of null value. If "Ignore Trailing Fields" check box is not checked, wherever the optional trailing field is of null value, padding should compulsorily be given for the same. It is to be noted that this option is applicable and is of use only if the following are satisfied:

The section should be a delimited one.

The trailing fields are set as optional, are of null value and no default value given for the same. (The trailing fields may be of Fixed Length, Length Preceded or Delimited)

## Default Field Delimiter

Here you can specify the default delimiter to be used for the delimited fields, if any, added within the Section under consideration. The field delimiter specified here is not applicable for Fixed Length or Length Preceded fields added within the section.

### Note:

A field delimiter can be of any length and can be a combination of both alphanumeric and special character.

The default field delimiter set at section level is applied only if there is no delimiter selected at field level ("Default" selected at field level). The delimiter

specified at field level overrides the default field delimiter specified at section level and the default field delimiter specified at section level overrides the one specified at Format level, i.e., if a Delimiter is set at Field level, that is used, if not, one set at Section Level is used (Provided you have selected "Default" from the Delimiter combo-box under the Field Type tab) or else, one set at the Format Level is used.

The default field delimiter specified here should not be the same as the section delimiter. Otherwise, the message would not be parsed.

The last field (delimited type) in a section is terminated by the section delimiter, if any specified and in the Input Message; you need not give the delimiter for such a field.

## Quote

The Quote combo box provided under **Default Field Delimiter** is used to specify the escape sequence to be used in case the field delimiter character/s is/are to be included in the value of the field. Refer [Quote explained in the section "Adding a Delimited field"](#) for more details.


## See Also:

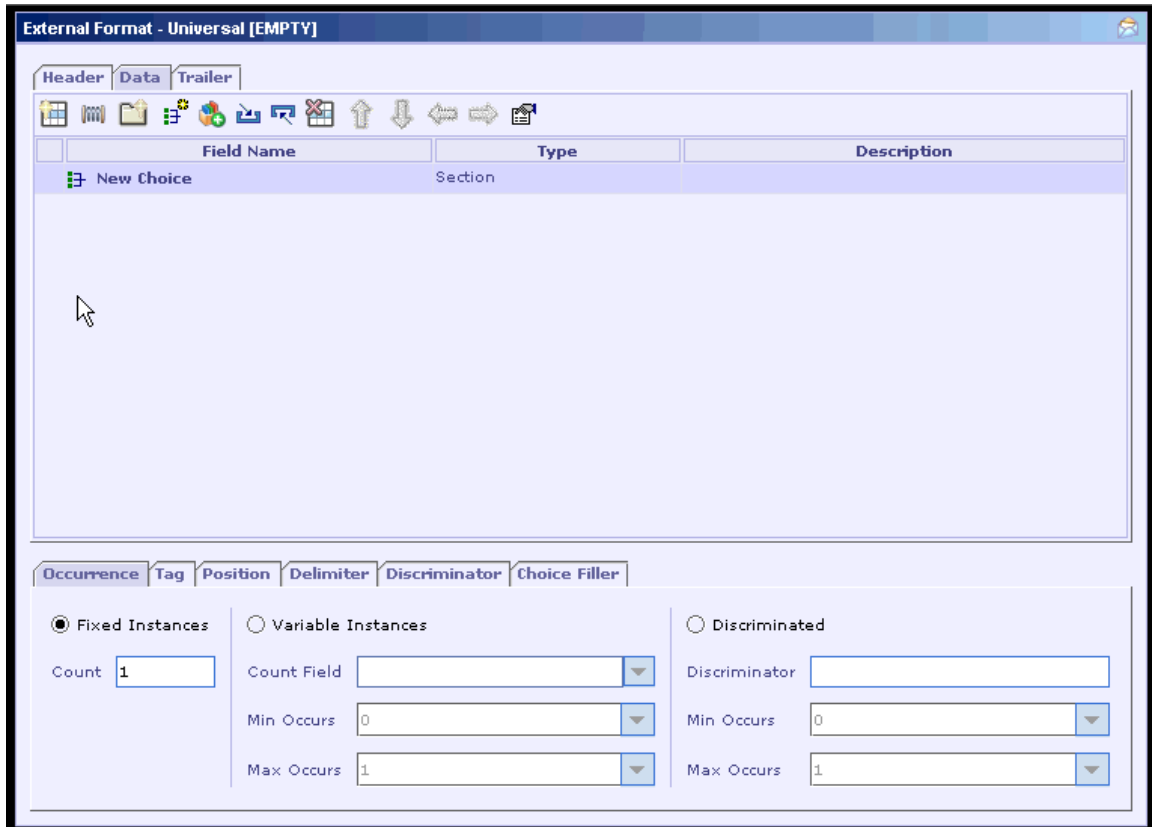
[Adding a Sequence type section and specifying properties](#)

## Adding a *Choice* Type Section and Specifying Properties

In a Choice section, only one of the child elements may occur in the message. So, if you want only one of the child elements of a section to appear in input/output based on some identification, you can make use of a Choice section (i.e. why it is named as "Choice"). A Choice can have as its child elements another Choice / Field / Filler / Sequence / All. In a Choice, all child elements have the same starting position as only one of them may occur. There cannot be duplicate Choices (Choices with same name) at the same level. In a Choice section, all child elements except one must be either self-discriminating (for e.g. tagged) or should have explicit discriminator to uniquely identify them.

### To add a Choice section in the Universal - External Format UI

1. Click the **Add Choice** button  in the Universal - External Format UI.
2. A new row (Choice) is added in the External Format UI with default name 'New Choice and default type set as 'Section' as shown below.



You can change the Choice name as per your requirement by double-clicking on the column under "Field Name". Note that there cannot be duplicate Choices (Choices with same name) at the same level.

- Under the column **Alias** (which is hidden in the above picture), you can specify the alias name (substitute name) to be used for the Choice when specifying a Validation/Mapping formula involving that Choice. This property is optional. (If a Choice is assigned an alias name, the qualified name of that Choice is replaced with its newly assigned alias name in all the places it is used.)
- Under the column **Description**, you can mention any description for the added Choice. This property is also optional.

## Specifying properties for a Choice

The Choice properties panel has six tabs,

Property Tab	Usage
Occurrence	Can specify the number of times the Choice occurs in the message & find out the presence of an optional choice
Tag	Can specify the tag and separator for the Choice
Position	Displays the position of the Choice in the message, data and its length
Delimiter	Can specify the Delimiter for the Choice as well as the Default Field Delimiter (for the fields added under it)
Discriminator	Can specify the Discriminator for the child elements of the Choice
Choice Filler	Can make all the child elements to be of equal length by padding

### Occurrence tab:

This property is equally applicable for any section. Refer [‘Occurrence’ tab](#) explained above under Chapter 4.2.3.

### Tag tab:

This property is equally applicable for any section. Refer [‘Tag’ tab](#) explained above under Chapter 4.2.3.

### Position tab:

This property is equally applicable for any section. Refer [‘Position’ tab](#) explained above under Chapter 4.2.3.

### Delimiter tab:

This property is equally applicable for any section. Refer [‘Delimiter’ tab](#) explained above under Chapter 4.2.3.

**Discriminator tab:**

Under this tab, you can specify the Discriminator that can be used to uniquely identify each of the child elements of a Choice section. (Discriminating the child elements becomes necessary as only one of them can occur). Note that this Discriminator has nothing to do with the Discriminator that is displayed under the Occurrence tab. (That Discriminator is used to detect the presence of an optional section while this Discriminator is used to identify the child elements of the Choice)

Field	Discriminator
F1	
F2	
F3	

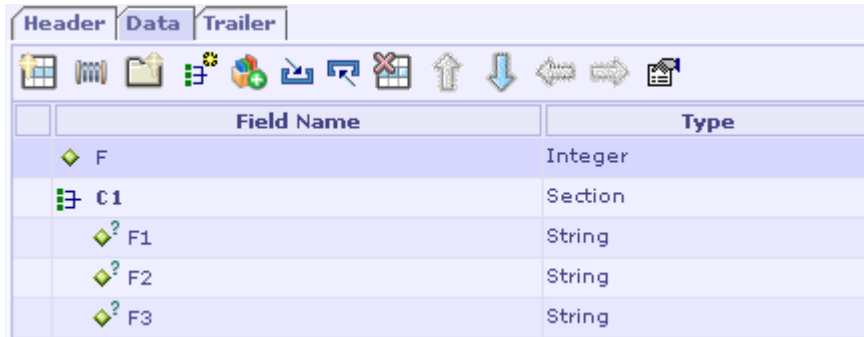
As seen in the above picture, under the **Discriminator** tab, the child elements of the selected Choice are automatically listed under the column **Field** and against each child element, you can enter the required Discriminator under **Discriminator** column. The Discriminator specified should be a Boolean formula (should return true or false value), which can make use of all the preceding fields and can also peek at the rest of the record/message and return true if it detects the presence of the corresponding child element in the input. (Note that when the cursor is in a cell under the Discriminator column, the formula icon is displayed in that cell, "Edit Formula" icon is enabled in the toolbar and Edit Formula hyperlink is displayed in the Status Bar). The various types of Discriminator that can be specified are listed below:

**Types of Discriminator (*Choice Section*)****Discriminate based on the value of one or more preceding fields**

You can set a field or more than one field as Discriminator field(s) and give Discriminator formula for the child elements of the Choice based on the value of the field(s). The field(s) must have unique values to represent each of the choices. The field(s) should necessarily appear before the point where decision has to be made.

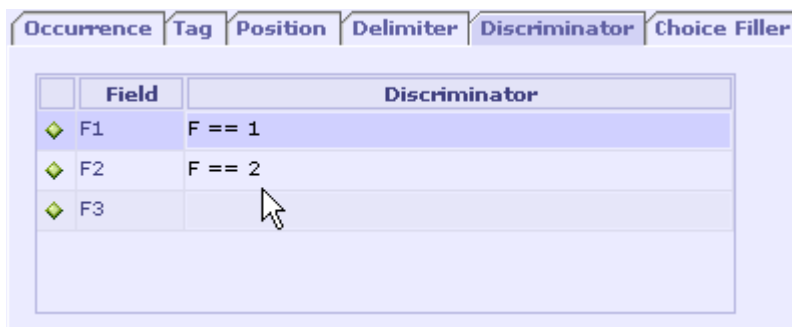
**E.g. 1. – Based on one preceding field**

Consider the following message structure.



Field Name	Type
F	Integer
C1	Section
F1	String
F2	String
F3	String

Here, we can use the integer field F as a Discriminator field and based on its value, specify Discriminator for the child elements F1 and F2 (We are not specifying Discriminator for F3 as at most one child element of a choice need not have Discriminator) for the Choice C1 as shown below:



Field	Discriminator
F1	F == 1
F2	F == 2
F3	

Here, if the value of the field F is 1, then the child element F1 should appear in input. If its value is 2, then the child element F2 should appear. Else, child element F3 should appear.

**Note:**

The conditions are executed in the order of the child elements and one that matches first will be used. It is your responsibility to ensure that there are not conflicts.

You can also choose to specify a discriminator for all the fields (instead of skipping one). In this case, if all the conditions fail, an exception is raised.

**E.g. 2. – Based on more than one preceding field**

Consider the following message structure.

Field Name	Type
A	Integer
B	Integer
C	Integer
C1	Section
F1	String
F2	String
F3	String

Here, we can make use of all the three preceding integer fields A, B and C (A & B of Fixed Length 1 & C of Fixed Length 2) as Discriminator fields and based on their value, specify Discriminator for the child elements F1 and F2 for the Choice C1 as shown below:

Field	Discriminator
F1	$C < A+B$
F2	$C == A+B$
F3	

Here, if the value of the field C is less than the sum of the values of the fields A and B, then the child element F1 should appear in input. If the value of C is equal to the sum of the values of A and B, then the child element F2 should appear. Else, child element F3 should appear. (A = 1, B = 3, C = 02, implies F1 should appear; A = 4, B = 6, C = 10 implies F2 should appear; Else F3 should appear)

**Discriminate by peeking into the message**

By using the Universal Plug-in specific peek functions, the user can write a Discriminator formula that looks at the rest of the input message/enclosing record to ascertain the presence of the child element.



E.g. Consider the following message structure:

Header		Data	Trailer
	Field Name		Type
◆	F		Integer
◆	C1		Section
◆	F1		String
◆	F2		String
◆	F3		String

Let F1, F2 & F3 be Fixed Length fields of Length 1. Here, we can make use of the peek function to discriminate the child elements as shown below:

Occurrence	Tag	Position	Delimiter	Discriminator	Choice Filler
◆	F1			Peek(1) == "A"	
◆	F2			Peek(1) == "B"	
◆	F3				

**Peek(1)** returns 1 character from the current parser location. (Refer [Universal Plug-in Peek Formula Functions](#)). So, if the character returned is "A", the child element F1 should appear in the input. If it is "B", then the child element F2 should appear. Else, the child element F3 should appear.

**Note:**

Discriminator formulae are used only during parsing. They have no role to play on the output side.

**Automatic Discrimination**

If each of the child elements of the Choice (except one) has the following attributes specified, then that would act as an automatic discrimination and you need not specify a Discriminator formula for the child elements in that case.

Is either tagged (or)

Has a Fixed value filler added to it at a fixed location, i.e. as the topmost field or preceded by fixed length fields (applicable in the case where the child element is a section) (or)

Is of Discriminated type (Occurrence marked as Discriminated) with Discriminator formula specified under the 'Discriminated' property under the 'Occurrence' tab. (applicable in the case where the child element is a section)

(From the above it is evident that the child elements of a Choice can be specified with different types of discriminators. For e.g., let there be four child elements for a Choice. You can specify a Tag for one child element, add a fixed value filler for one (applicable if the child element is a section), mark one child element as Discriminated (applicable if the child element is a section) and specify a Discriminator formula under the 'Discriminated' property under the 'Occurrence' tab and leave one child element without specifying any discriminator.)

**Choice Filler tab:**

**Pad child elements to equal length**

The child elements of a Choice can be padded to be of the same length by checking the '**Pad to equal length**' check box available under this tab.



This property is applicable only if the length of all the child elements can be determined at design time itself, i.e., if the child elements are fields/fillers, they should be of fixed length and if they are sections, they should have only fixed length fields / fillers added under them. If the above constraints are not satisfied, the Designer would throw error message on validation.

For e.g., let there be a Choice 'C' with child elements as fields F1, F2 and F3 of Fixed Length and data type string with the following specification:

Field	Length	Tag
F1	3	A
F2	5	ABCD

F3	7	
----	---	--

If you set the **Pad to equal length** property to be true for the Choice C, then all the child elements would be automatically assigned the same length based on the length of the child element, which has the maximum length (including the length of the tag). In this case, lengths of the fields are: F1 : 3+1 = 4; F2 : 5+4 = 9; F3 : 7. So, all child elements would be assigned the maximum length, which is 9. While giving data, you have to ensure that you have given the appropriate padding. Let us suppose that the child element F1 occurs in the Choice. Then its value should be somewhat like "Aabc " (Space Padding given to fill the remaining 5 bytes)

#### See Also:


[Adding a Sequence type section and specifying properties](#)

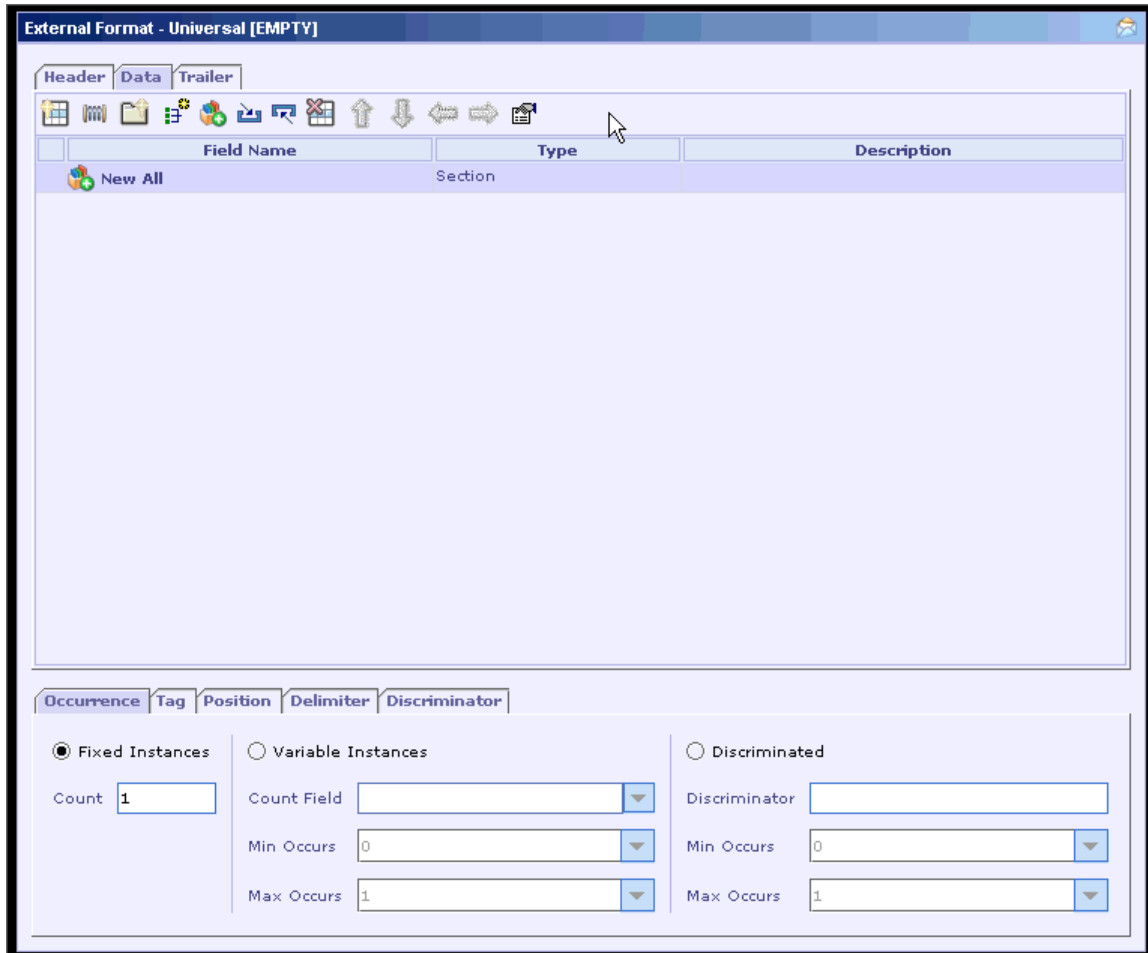
[Adding an All type Section and specifying properties](#)

## Adding an All type Section and specifying properties

In an 'All' type section, the child elements may occur in any order unlike a Sequence where they occur in the specified order. It can be called as an Unordered Section. In an 'All' type section, the order of occurrence of the child elements is not determined during design time, but only during run time, based on the discriminator specified for the child elements. Hence, in an 'All' type section, **all child elements must compulsorily have a Discriminator** (either tag or Fixed Value filler added in case of sections or Discriminator formula specified), unlike a Choice section where there can be at most one child element without Discriminator.

To add an All type section in the Universal - External Format UI

1. Click the **Add All (unordered section)** button  in the Universal - External Format UI.
2. A new row (All) is added in the External Format UI with default name 'New All' and default type set as 'Section' as shown below.



You can change the name of the section as per your requirement by double-clicking on the column under "Field Name". Note that there cannot be duplicate 'All' type sections ('All' type sections with same name) at the same level.

3. Under the column **Alias** (which is hidden in the above picture), you can specify the alias name (substitute name) to be used for the 'All' type section when specifying a Validation/Mapping formula involving that section. This property is optional. (If an 'All' type section is assigned an alias name, the qualified name of that section is replaced with its newly assigned alias name in all the places it is used.)
4. Under the column **Description**, you can mention any description for the added section. This property is also optional.

## Specifying properties for an 'All' type section

The properties panel for an 'All' type section has five tabs,

Property Tab	Usage
Occurrence	Can specify the number of times the 'All' type section occurs & find out the presence of an optional 'All' section
Tag	Can specify the tag and separator for the 'All' section
Position	Displays the position of the 'All' section in the message, data and its length
Delimiter	Can specify the Delimiter for the 'All' section as well as the Default Field Delimiter (for the fields added under it)
Discriminator	Can specify the Discriminator for the child elements of the 'All' section

### Occurrence tab:

This property is equally applicable for any section. Refer ['Occurrence' tab](#) explained above under Chapter 4.2.3.

### Tag tab:

This property is equally applicable for any section. Refer ['Tag' tab](#) explained above under Chapter 4.2.3.

### Position tab:

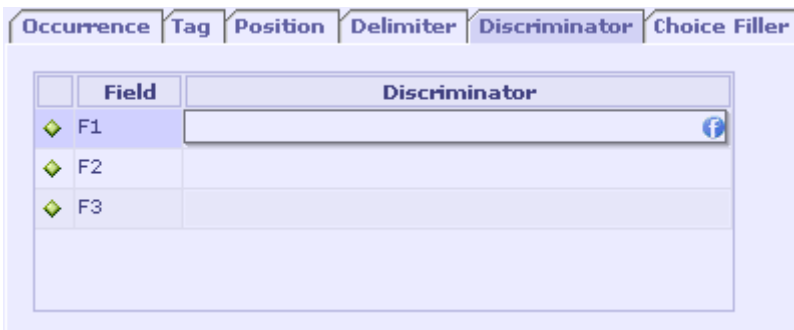
This property is equally applicable for any section. Refer ['Position' tab](#) explained above under Chapter 4.2.3.

### Delimiter tab:

This property is equally applicable for any section. Refer ['Delimiter' tab](#) explained above under Chapter 4.2.3.

### Discriminator tab:

Under this tab, you can specify the Discriminator that can be used to uniquely identify each of the child elements of an 'All' type section. (Discriminating the child elements of an 'All' type section becomes necessary to uniquely identify them as they can occur in any order). Note that this Discriminator has nothing to do with the Discriminator that is displayed under the Occurrence tab. (That Discriminator is used to detect the presence of an optional section while this Discriminator is used to uniquely identify the child elements of the 'All' type section)



As seen in the above picture, under the **Discriminator** tab, the child elements of the selected 'All' type section are automatically listed under the column **Field** and against each child element, you can enter the required Discriminator under **Discriminator** column. The Discriminator specified should be a Boolean formula (should return true or false value), which can make use of all the preceding fields and can also peek at the rest of the record/message and return true if it detects the presence of the corresponding child element in the input. (Note that when the cursor is in a cell under the Discriminator column, the formula icon is displayed in that cell, "Edit Formula" icon is enabled in the toolbar and "Edit Formula" hyperlink is displayed in the Status Bar). The types of Discriminator that can be specified for the child elements of an 'All' type section is as explained below.

## Types of Discriminator (All section)

### 1. Automatic Discrimination

If each of the child elements of the 'All' type section has any of the following attributes specified, then that would act as an automatic discrimination and you need not specify a Discriminator formula for the child elements in that case.

Is either tagged (or)

Has a Fixed value filler added to it at a fixed location, i.e. as the topmost field or preceded by fixed length fields (applicable in the case where the child element is a section) (or)

Is of Discriminated type (Occurrence marked as Discriminated) with Discriminator formula specified under the 'Discriminated' property under the 'Occurrence' tab. (applicable in the case where the child element is a section)

(From the above it is evident that the child elements of an 'All' type section can be specified different types of discriminators. For e.g., let there be three child elements for an 'All' type section. You can specify a Tag for one child element, add a fixed value filler for one (applicable if the child element is a section) and mark one child element as Discriminated (applicable if the child element is a section) and specify a Discriminator formula under the 'Discriminated' property under the 'Occurrence' tab)

## 2. Discriminate by peeking into the message

By using the Universal Plug-in specific peek functions, the user can write a Discriminator formula that looks at the rest of the input message/enclosing record to ascertain the presence of the child element.

E.g. Consider the following message structure:

Header		Data	Trailer
	<b>Field Name</b>		<b>Type</b>
	A 1		All
	F1		String
	F2		String
	F3		String

Let F1, F2 & F3 be Fixed Length fields of Length 1. Here, we can make use of the peek function to discriminate the child elements as shown below:

Occurrence	Tag	Position	Delimiter	Discriminator
				<b>Field</b>
				<b>Discriminator</b>
				F1
				Peek(1)=='A'
				F2
				Peek(1)=='B'
				F3
				Peek(1)=='C'

**Peek(1)** returns 1 character from the current parser location. (Refer [Universal Plug-in Peek Formula Functions](#)). So, if the character returned is "A", then the parser recognizes it as child element F1. If it is "B", then the parser recognizes it as child element F2. If it is "C", then the parser recognizes it as child element F3.

### 3. Discriminate based on the value of one or more preceding fields

Specifying a discriminator formula for the child elements based on the value of preceding fields is rare and uncommon. This is not simple as we saw in the case of a 'Choice' section, because, here more than one child element can occur and hence you cannot discriminate simply based on the single value of a preceding field.

**See Also:**

- [Adding a Sequence type section and specifying properties](#)
- [Adding a Choice type Section and specifying properties](#)

### A simple illustration of using *All* type section

Consider the following message sample:

Name=Dev;Age=25;A/c=Savings;

Let the specification for the message format be as under:

Section	Repeating for 5 times
Field Types	Delimited (Mandatory)
Delimiter	Semi-colon
Tagged (Yes/No)	Yes
Ordered/Unordered	Unordered

We can represent the above format by adding an 'All' type section and adding three fields as its child elements as shown below:





The specification and properties for the message format can be entered as under:

Under the **Occurrence** tab for the 'All' section 'A', check **Fixed Instances** radio button and give **Count** as 5.

For all the three fields viz., NAME, AGE & A/C, under the **General** tab, check the **Required** check box.

For all the three fields, under the **Field Type** tab, check the **Delimited** radio-button and specify the **Delimiter** as ; (semi-colon)

Under the **Tag** tab, specify the tag for the fields as under:

Field	Tag
NAME	Name=
AGE	Age=
A/C	A/c=

## To enter a Fix message format using an *All* type section

In a Fix message format, the fields/sections added are in the form of Name-Value pairs, i.e., they should compulsorily have a tag and they can appear in any order. Hence, you can represent a Fix message format using Universal Plug-in by adding an 'All' type section as the topmost element and adding the Fix message format with its fields/sections as the child elements of the 'All' type section and giving tag for all the child elements.


### See Also:

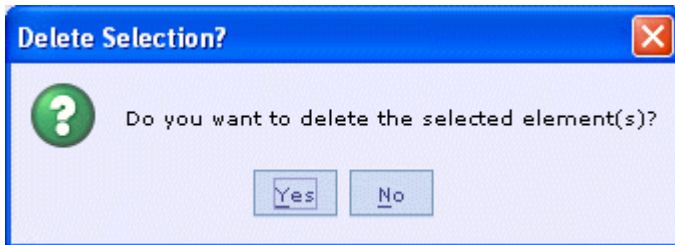
[Adding an All type Section and specifying properties](#)

[Entering a Universal message](#)

[Removing a Field/Section](#)

## Removing a Field/Section

Select the field(s)/section(s) that is to be removed. Click the  button in the toolbar. The following dialog will be displayed.



Select 'Yes' to delete the selected elements.

You can also select the field(s)/section(s) to be removed and press the 'Delete' key. The Delete Selection dialog will be displayed. Click 'Yes' to delete the selected elements.

### See Also:

[Adding Fields and specifying properties](#)

[Adding Fillers and specifying properties](#)

[Adding a Sequence type section and specifying properties](#)

[Adding a Choice type Section and specifying properties](#)

[Adding an All type Section and specifying properties](#)

## Universal Plug-in Peek Formula Functions

Universal Plug-in Peek formula functions are available in the input side of the Universal format plug-in. They are used to peek at the input and decide on how the parsing should proceed. All functions are applied relative to the current parser location. Along with these functions that depend on unparsed input, you can directly refer to fields that have already been parsed (fields above the field for which you are specifying the formula).

Function	Description
<code>Peek(int start, int length)</code>	Returns 'length' characters as string from the location specified by 'start', relative to current parser location.
<code>Peek(int length)</code>	Returns 'length' characters from the current

	parser location.
<code>PeekRest()</code>	Returns the rest of the characters in the current record
<code>IsNext(String str)</code>	Matches the next available input (during parsing) with the string 'str' and returns true if it starts with the specified string. This is equivalent to  <code>Peek(Length(str)) == str</code>

You can use them as part of,

Discriminator formula for sections

Field discriminators of a Choice/All type section

Length preceded field formula

Variable instances section count formula

The ability to perform *context dependent parsing* (as in peek functions) is a distinguishing feature of Universal plug-in.

**See Also:**

[Adding a Choice type Section and specifying properties](#)

[Adding an All type Section and specifying properties](#)

[Adding a Sequence type section and specifying properties](#)

[Adding Fields and specifying properties](#)

## Save Selection As Template / Add Template

Whole or part of a Universal message format can be saved as a template in some location and can be added in any other Universal message format.


**See Also:**

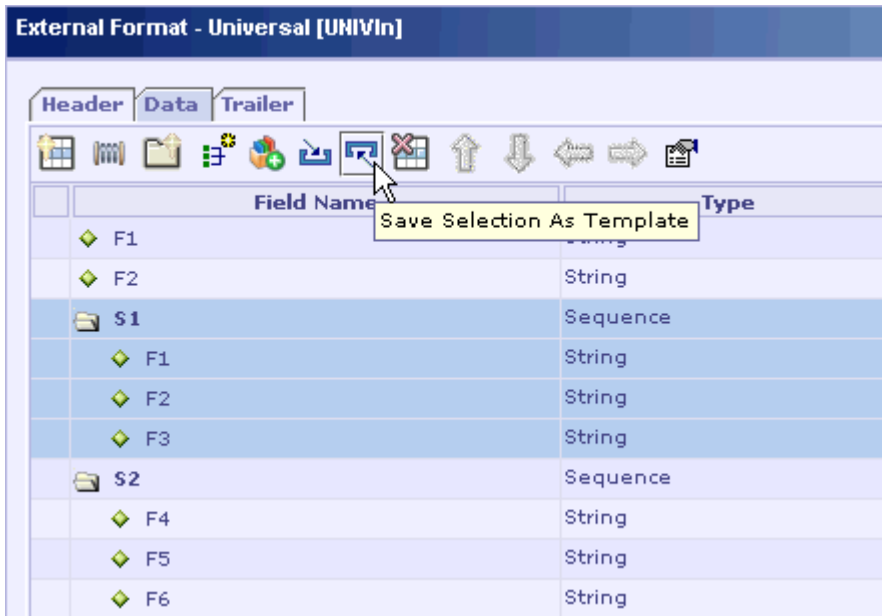
[Save As Template](#)

[Add Template](#)

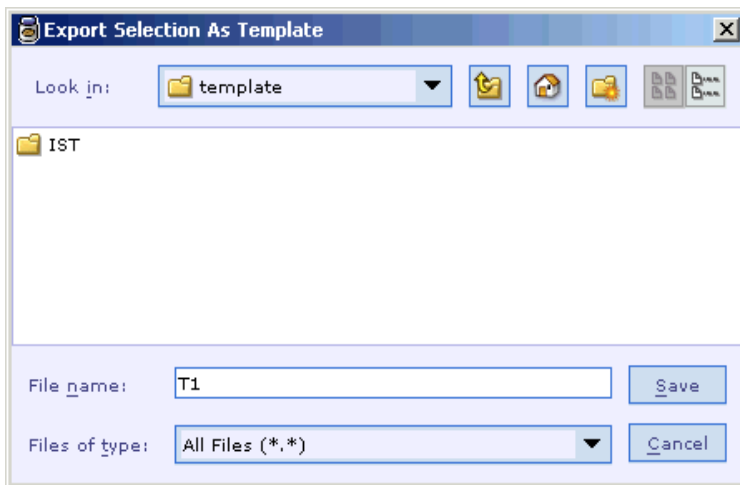
[Saving a Universal Message Format](#)

## Save As Template

To save some portion of a Universal Message format as a template, select the required portion and click the **Save Selection as Template** button  in the Universal - External Format UI.




In the **Export Selection As Template** window that appears, specify the required location and file name for storing the template and click 'Save' button. The selection will be saved in the given location as a template file. Note that if you select only some of the child elements of a section and save the selection as template, by default, the whole section is saved as template.

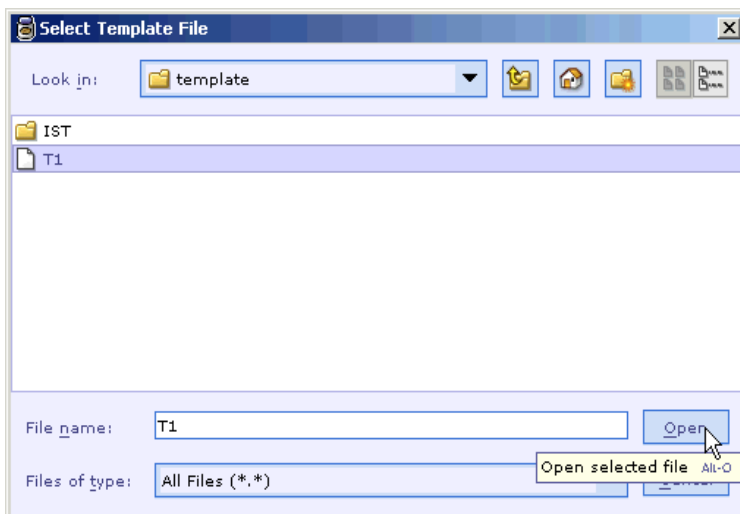


**See Also:**[Add Template](#)[Saving a Universal Message Format](#)

## Add Template

To add a template (saved in some location) in a Universal Message format, select the location in the table of the UI where the template is to be added and click the **Add Template** button  in the Universal - External Format UI.

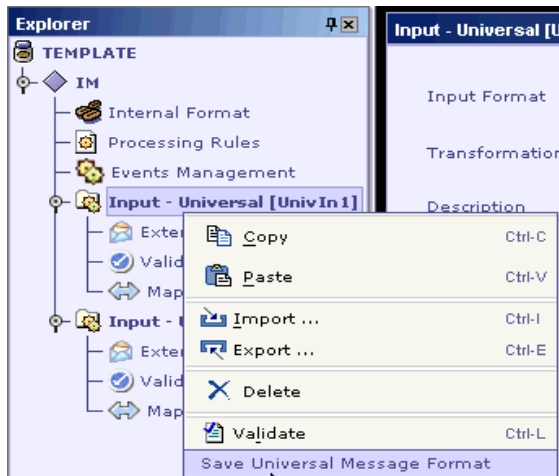
In the **Select Template File** window that appears, select the required template from the required location and click 'Open' button. The selected template will be added in the Universal message format.

**See Also:**[Save As Template](#)

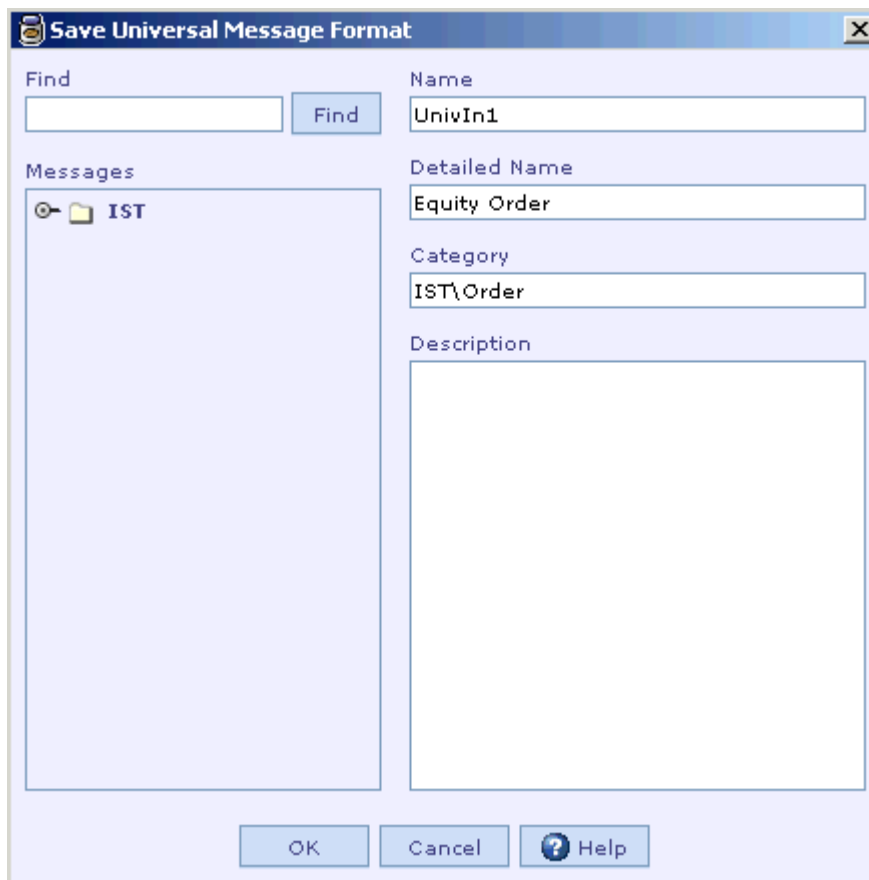
## Saving a Universal Message Format

A Universal message format once fully entered in Designer can be saved in a default standard location so that it is available for creating other formats based on it.

1. To save an entered Universal message format in the standard location, right-click the Universal format node in the **Explorer** and choose the context menu **Save Universal Message Format**.



2. The **Save Universal Message Format** dialog appears. Enter the **Name**, **Detailed Name**, **Category** and **Description**, if any for the format. The category entered can be an existing one, or a new category. Click OK. (You can also overwrite any of the Existing Formats with the current one by selecting the same from the Existing Formats panel.)





3. The Universal message format is saved as XML file in the location <installation dir>\config\Universal\messages in the given **Format Name**.
4. Universal Formats saved this way are available for creating formats in future.  
Refer [Creating a Universal Format based on an existing Universal format](#)

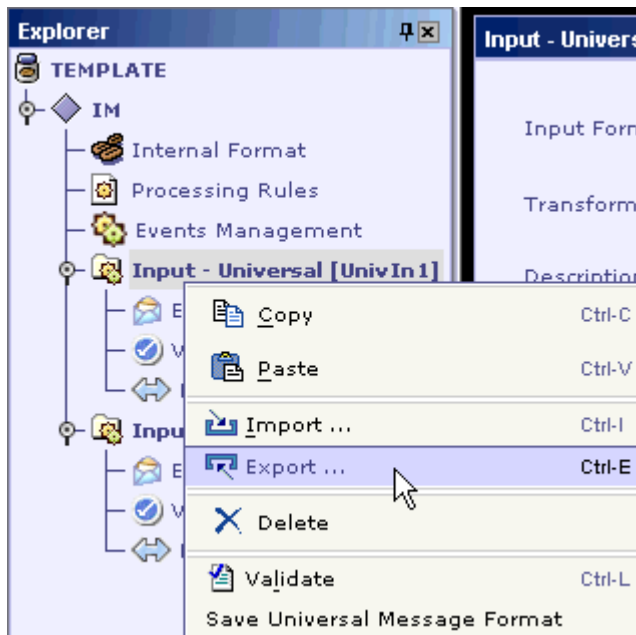
**See Also:**

[Creating an empty Universal message Format](#)  
[Exporting a Universal Message Format](#)  
[Importing a Universal Message Format](#)  
[Save Selection As Template / Add Template](#)

## Exporting a Universal Message Format

Universal message format can also be saved using the usual export method, allowing to save the message in XML format.

1. To save a Universal format, right-click the Universal format node in the **Explorer** and choose the context menu **Export....**

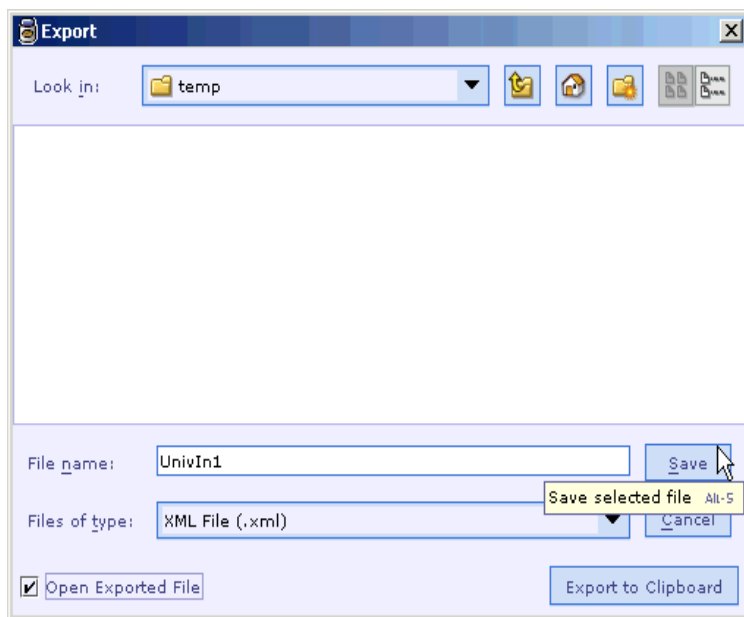


2. In the **Export** dialog that appears, select the location to save the exported file. Select the file type in the **Files of type** listbox and enter a name to save the file in the **File name** text field. Click **Save** button.

### Note:

You also have the option of exporting the file (in the format you have chosen) to the Clipboard by clicking the **'Export to Clipboard'** button in the dialog. If you want to simultaneously open the exported file, you can check the **'Open Exported File'** checkbox.





3. The Universal message format is saved in the specified location with the file name mentioned.

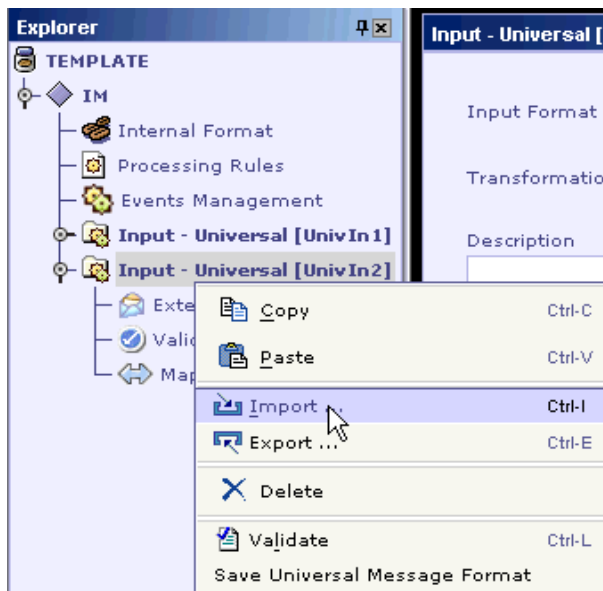
**See Also:**

[Importing a Universal Message Format](#)  
[Importing COBOL copy book structure](#)

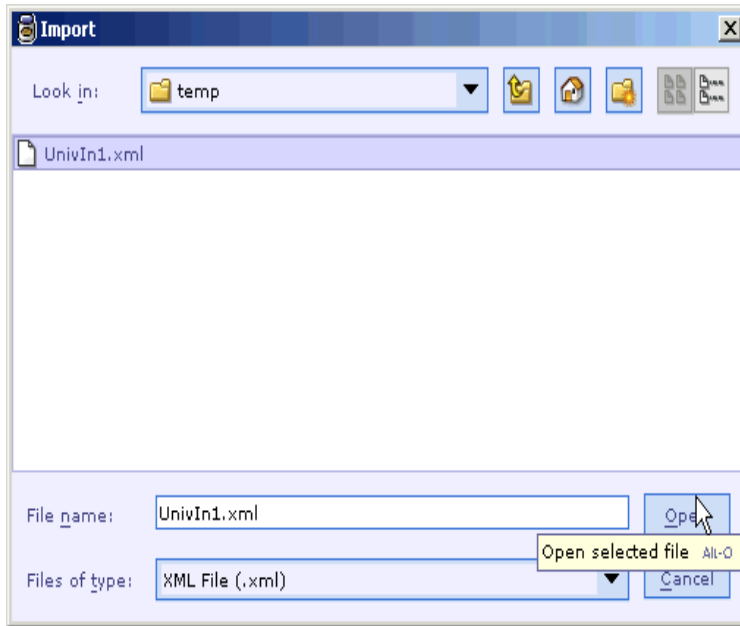
## Importing a Universal Message Format

Universal message format saved using the export method, can be imported in any other Universal message format.

1. To import a Universal format, right-click the Universal format node (where you want to import a message) in the **Explorer** and choose the context menu **Import...**



2. In the **Import** dialog that appears, select the file to be imported. Click **Open** button.



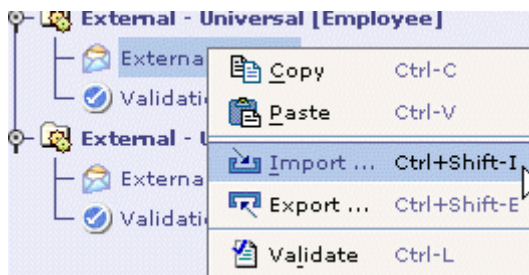
The selected message format in the file is imported in the current node.

**See Also:**

- [Exporting a Universal Message Format](#)
- [Importing COBOL copy book structure](#)

## Importing COBOL copy book structure

To import a COBOL copy book structure right click the 'External Format' node and select 'Import' menu item.



In the 'Import' dialog that appears select 'CBL File' as file type to be displayed. Select the file that needs to be imported. The CBL will be imported as Universal message.

When a copybook with multiple top-level items is imported, all of them are imported as separate sections. User may have to delete sections that are not needed.

The manner in each item in the .CBL file is imported is discussed below.

## COBOL Copy Book to Universal Mapping

### Fields

The data type of the field is set based on the picture character settings.

PIC X – String

PIC 9 – Integer

PIC 9(13)V9(2) – Double

If COMP-3 clause is specified for an item then the encoding of the field is set as 'Packed Decimal'.

If COMP-1, COMP-2 or COMP-5 clause is specified for an item, then the encoding of the field is set as 'Binary'.

If the 'VALUE' clause is specified then the value is set as the default value for the field.

If 'SIGN SEPARATE' clause is specified in a picture character setting for an item, the length of the field will be the length specified in the picture setting + an additional digit for the sign character.

If the item name is FILLER then it is created as a FILLER field.

CBL Item	Universal Field
PIC-TEST-1 PIC X.	String field of length 1
PIC-TEST-3 PIC X(3).	String field of length 3
20 PIC-TEST-4 PIC S9.	Integer field of length 1
SALARY PIC 9(5)V(2).	Double field of length 7. The format of the field is 5.2. The decimal point option is none (i.e.) field does not have any decimal point.

SALARY PIC 9(5)V(2) SIGN LEADING SEPARATE.	Double field of length 8. Format of the field is set as 6.2. The additional length is because the sign character is treated as separate.
VALUE-TEST-5 PIC XXXXX VALUE "HELLO".	The field created is of type 'String' and has default value 'HELLO'.
CLIENT1 PIC 9(03) COMP-3.	The field created is of type Integer with Packed Decimal encoding. The actual length of the field (Digits/Formats) is 3. The length of the field (in bytes) in the message is 2 (digits+1/2. 3+1/2).
CLIENT2 PIC 9(03) COMP-1.	The field created is of type Integer with Binary encoding. The length of field is 3.
CLIENT2 PIC 9(03) COMP-2.	The field created is of type Integer with Binary encoding. The length of field is 3.
CLIENT2 PIC 9(03) COMP-5.	The field created is of type Integer with Binary encoding. The length of field is 3.
FILLER PIC X(60)	Field created is of type 'FILLER' of length 60.

### Sections:

If a Copybook item has OCCURS clause or does not have PIC mentioned in it the entry is represented as section in the Universal Message. The entries within are imported as fields/sub-sections.

If the OCCURS clause is not mentioned and the entry does not have PIC mentioned then it is created as fixed instance section with value 1.

If the OCCURS clause is specified without 'DEPENDING ON' clause the entry is created as a fixed instance section with instance value set as the value specified in the OCCURS CLAUSE.

If the OCCURS clause is specified with 'DEPENDING ON' clause the entry is created as a 'Variable Instances' section. The 'Count Field' for the section is the field specified in the DEPENDING ON clause. The min/max occurs are set based on the values specified in the OCCURS clause.

CBL Item	Universal Section
02 TRANSACTION-REFERENCE-NUMBER.	Fixed instance section with instance value 1.
02 STATEMENTLINE OCCURS 100 TIMES.	Fixed instance section with instance value 100.
02 OPENING-BALANCE-COUNT OCCURS 0 TO 100 TIMES DEPENDING ON STATEMENT-NUMBER-COUNT.	Variable instances section with count field as 'STATEMENT-NUMBER-COUNT'. Min occurs is 0 and max occurs is 100.
02 STATEMENT-NUMBER OCCURS 0 TO 100 TIMES DEPENDING ON STATEMENT-NUMBER-COUNT. 03 STATEMENT-NUMBER PIC 9(5). 03 OPENING-BALANCE OCCURS 1 TO 20 TIMES DEPENDING ON OPENING-BALANCE-COUNT. 04 D-C-MARK PIC X(1).	A repeating section Statement Number is created. It has a field Statement number and a repeating inner section Opening Balance.

**Note:**

Any comment specified in the copybook source is set as the section item's description while importing.

While importing the copybook source, columns excluding 6-72 are excluded. Contents in columns 1-5 and beyond column 72 in the copybook file are not imported.

A copybook source is shown and the Universal representation of the source is shown below

**Copybook Source:**

```
01 STATEMENT-MESSAGE.
    02 TRANSACTION-REFERENCE-NUMBER    PIC X(16).
```

```

02 ACCOUNT-IDENTIFICATION PIC X(35).
02 STATEMENT-NUMBER.
    03 STATEMENT-NUMBER PIC 9(5).
    03 SEQUENCE-NUMBER PIC 9(5).
02 OPENING-BALANCE.
    03 D-C-MARK PIC X(1).
    03 DATE-FLD PIC 9(6).
    03 CURRENCY PIC X(3).
    03 AMOUNT PIC 9(13)V9(2).
* STATEMENT LINE
02 STATEMENTLINE.
    03 VALUE-DATE PIC X(6).
    03 ENTRY-DATE PIC 9(4).
    03 DEBIT-CREDIT-MARK PIC X(2).
* FUNDS CODE FOR STATEMENT
    03 FUNDS-CODE PIC X(1).
    03 AMOUNT PIC 9(13)V9(2).
    03 TRANSACTION-TYPE-IDENTIFICATIO PIC X(4).
    03 REFERENCE-FOR-THE-ACCOUNT-OWNE PIC X(16).
    03 ACCOUNT-SERVICING-INSTITUTIONC PIC X(16).
    03 SUPPLEMENTARY-DETAILS PIC X(34).

```

## Universal Representation:

Header			Data	Trailer	
Field Name	Type	Description			
TRANSACTION-REFERENCE-NUMBER	String				
ACCOUNT-IDENTIFICATION	String				
<b>STATEMENT-NUMBER</b>	Sequence				
STATEMENT-NUMBER	Integer				
SEQUENCE-NUMBER	Integer				
<b>OPENING-BALANCE</b>	Sequence				
D-C-MARK	String				
DATE-FLD	Integer				
CURRENCY	String				
AMOUNT	Double				
<b>STATEMENTLINE</b>	Sequence	* STATEMENT LINE			
VALUE-DATE	String				
ENTRY-DATE	Integer				
DEBIT-CREDIT-MARK	String				
FUNDS-CODE	String	* FUNDS CODE FOR STATEMENT			
AMOUNT	Double				
TRANSACTION-TYPE-IDENTIFICATIO	String				
REFERENCE-FOR-THE-ACCOUNT-OWNE	String				
ACCOUNT-SERVICING-INSTITUTIONC	String				
SUPPLEMENTARY-DETAILS	String				

**See Also:**

[Exporting a Universal Message Format](#)

[Importing a Universal Message Format](#)



## APPENDIX

The following table lists the Escape sequences in strings that are applicable while specifying values for the following:

1. Delimiter
2. Terminator
3. Default Value
4. Fixed Value
5. Tag

Escape Sequence	Remark
\n	= ASCII 13
\r	
\t	
\b	
\f	
\"	
\'	
\\	
\#x	To represent Hexa-Decimal character

**See Also:**

[Creating a Universal Format](#)

[Universal Format Options](#)

[Entering a Universal message](#)

[Saving a Universal Message Format](#)