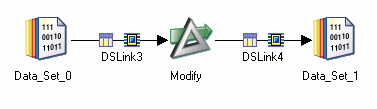
The Modify stage alters the record schema of its input data set. The modified data set is then output. You can drop or keep columns from the schema, or change the type of a column.

The Modify stage is a processing stage. It can have a single input link and a single output link.

The Modify stage alters the record schema of its input data set. The modified data set is then output. You can drop or keep columns from the schema, or change the type of a column.



The stage editor has three pages:

* [**Stage Page**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Stage_Page_modify_stage.html?view=kc). This is always present and is used to specify general information about the stage.
* [**Input Page**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Input_Page_modify_stage.html?view=kc). This is where you specify details about the input link.
* [**Output Page**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Outputs_Page_modify_stage.html?view=kc). This is where you specify details about the modified data being output from the stage.

Although the Modify stage is classified as a processing stage, it does not behave in the same way as other processing stages. In the Monitor window of the Director client, the stage does not show processing rows, and the stage does not show an outgoing row count in the Designer client.

The following example takes a data set with columns.

The following example takes a data set comprising the following columns:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **SQL Type** | **Length** | **Scale** |
| CUSTID | Decimal | 6 |  |
| NAME | Char | 45 |  |
| ADDRESS | Char | 40 |  |
| CITY | Char | 30 |  |
| STATE | Char | 2 |  |
| ZIP | Char | 9 |  |
| AREA | Decimal | 3 |  |
| PHONE | Char | 9 |  |
| REPID | Decimal | 4 |  |
| CREDITLIMIT | Decimal | 9 | 2 |
| COMMENTS | VarChar | 254 |  |

The modify stage is used to drop the REPID, CREDITLIMIT, and COMMENTS columns. To do this, the stage properties are set as follows:

Specification = DROP REPID, CREDITLIMIT, COMMENTS[Copy](javascript:void(0);)

The easiest way to specify the outgoing meta data in this example would be to use runtime column propagation. You could, however, choose to specify the meta data manually, in which case it would look like:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **SQL Type** | **Length** | **Scale** |
| CUSTID | Decimal | 6 |  |
| NAME | Char | 45 |  |
| ADDRESS | Char | 40 |  |
| CITY | Char | 30 |  |
| STATE | Char | 2 |  |
| ZIP | Char | 9 |  |
| AREA | Decimal | 3 |  |
| PHONE | Char | 9 |  |

You could achieve the same effect by specifying which columns to keep, rather than which ones to drop. In the case of this example the required specification to use in the stage properties would be:

KEEP CUSTID, NAME, ADDRESS, CITY, STATE, ZIP, AREA, PHONE

You could also change the data types of one or more of the columns from the above example.

Say you wanted to convert the CUSTID from decimal to string, you would specify a new column to take the converted data, and specify the conversion in the stage properties:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **SQL Type** | **Length** | **Scale** |
| conv\_CUSTID | Char | 20 |  |
| NAME | Char | 45 |  |
| ADDRESS | Char | 40 |  |
| CITY | Char | 30 |  |
| STATE | Char | 2 |  |
| ZIP | Char | 9 |  |
| AREA | Decimal | 3 |  |
| PHONE | Char | 9 |  |
| REPID | Decimal | 4 |  |
| CREDITLIMIT | Decimal | 9 | 2 |
| COMMENTS | VarChar | 254 |  |

Specification = conv\_CUSTID = CUSTID[Copy](javascript:void(0);)

Some data type conversions require you to use a transform command, a list of these, and the available transforms, is given in ["Specification"](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/r_deeref_Options_Category_modify_stage.html?view=kc) . The decimal to string conversion is one that can be performed using an explicit transform. In this case, the specification on the Properties page is as follows:

conv\_CUSTID:string = string\_from\_decimal(CUSTID)[Copy](javascript:void(0);)

You can also use the Modify stage to handle columns that might contain null values.

Any of the columns in the example, other than CUSTID, could legally contain a null value. You could use the modify stage to detect when the PHONE column contains a null value, and handle it so no errors occur. In this case, the specification on the Properties page would be:

PHONE = handle\_null (PHONE,-128)[Copy](javascript:void(0);)

Other null handling transforms are described in ["Specification"](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/r_deeref_Options_Category_modify_stage.html?view=kc) .

This section specifies the minimum steps to take to get a Modify stage functioning.

**About this task**

InfoSphere® DataStage® has many defaults which means that it can be very easy to include Modify stages in a job. InfoSphere DataStage provides a versatile user interface, and there are many shortcuts to achieving a particular end, this section describes the basic method, you will learn where the shortcuts are when you get familiar with the product.

To use a Modify stage:

* In the Stage page [**Properties Tab**](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/r_deeref_Properties_Tab_modify_stage.html?view=kc), supply the Modify specification.
* Ensure you have specified the meta data for the input and output columns

You can specify aspects of the Modify stage from the Modify stage: Stage page.

The General tab allows you to specify an optional description of the stage. The Properties tab lets you specify what the stage does. The Advanced tab allows you to specify how the stage executes.

Use the Properties tab to specify how the Modify stage operates.

The Properties tab allows you to specify properties which determine what the stage actually does. The modify stage only has one property, although you can repeat this as required.

| **Category/Property** | **Values** | **Default** | **Mandatory?** | **Repeats?** | **Dependent of** |
| --- | --- | --- | --- | --- | --- |
| Options/[Specification](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/r_deeref_Options_Category_modify_stage.html?view=kc) | string | N/A | Y | Y | N/A |
| *Table 1. Properties* | | | | | |

Use the Options category to specify how the Modify stage operates.

**Specification**

This is a statement with one of the following the forms:

* DROP *columnname* [, *columnname*]
* KEEP *columnname* [, *columnname*]
* *new\_columnname*:*new\_type* = [*explicit\_conversion\_function*] *old\_columnname*

If you choose to drop a column or columns, all columns are retained except those you explicitly drop. If you chose to keep a column or columns, all columns are excluded except those you explicitly keep.

If you specify multiple specifications each will be carried out sequentially.

Some type conversions InfoSphere® DataStage® can carry out automatically, others need you to specify an explicit conversion function. Some conversions are not available.

The following table summarizes the availability, with the source fields shown vertically and the target fields shown horizontally. A value of "d" indicates automatic (default) conversion, "m" indicates that manual conversion is required, a blank square indicates that conversion is not possible:

|  | **int8** | | **uint8** | **int16** | | **uint16** | | **int32** | **uint32** | **int64** | **uint64** | **sfloat** | **dfloat** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **int8** |  | | d | d | | d | | d | d | d | d | d | d m |
| **uint8** | d | |  | d | | d | | d | d | d | d | d | d |
| **int16** | d m | | d |  | | d | | d | d | d | d | d | d |
| **uint16** | d | | d | d | |  | | d | d | d | d | d | d |
| **int32** | d m | | d | d | | d | |  | d | d | d | d | d |
| **uint32** | d | | d | d | | d | | d |  | d | d | d | d |
| **int64** | d m | | d | d | | d | | d | d |  | d | d | d |
| **uint64** | d | | d | d | | d | | d | d | d |  | d | d |
| **sfloat** | d m | | d | d | | d | | d | d | d | d |  | d |
| **dfloat** | d m | | d | d | | d | | d | d | d | d | d |  |
| **decimal** | d m | | d | d | | d | | d m | d | d m | d m | d | d m |
| **string** | d m | | d | d m | | d | | d | d m | d | d | d | d m |
| **ustring** | d m | | d | d m | | d | | d | d m | d | d | d | d m |
| **raw** | m | |  |  | |  | | m |  |  |  |  |  |
| **date** | m | |  | m | |  | | m | m |  |  |  |  |
| **time** | m | |  |  | |  | | m |  |  |  |  | m |
| **time stamp** | m | |  |  | |  | | m |  |  |  |  | m |
|  | | **decimal** | | | **string** | | **ustring** | | **raw** | **date** | **time** | **timestamp** | |
| **int8** | | d | | | d m | | d m | |  | m | m | m | |
| **uint8** | | d | | | d | | d | |  |  |  |  | |
| **int16** | | d | | | d m | | d m | |  |  |  |  | |
| **uint16** | | d | | | d m | | d m | |  |  |  |  | |
| **int32** | | d | | | d m | | d m | |  | m |  | m | |
| **uint32** | | d | | | m | | m | |  | m |  |  | |
| **int64** | | d | | | d | | d | |  |  |  |  | |
| **uint64** | | d | | | d | | d | |  |  |  |  | |
| **sfloat** | | d | | | d | | d | |  |  |  |  | |
| **dfloat** | | d m | | | d m | | d m | |  |  | m | m | |
| **decimal** | |  | | | d m | | d m | |  |  |  |  | |
| **string** | | d m | | |  | | d | |  | m | m | m | |
| **ustring** | | d m | | | d | |  | |  |  | m | m | |
| **raw** | |  | | |  | |  | |  |  |  |  | |
| **date** | |  | | | m | | m | |  |  |  | m | |
| **time** | |  | | | m | | m | |  |  |  | d m | |
| **timestamp** | |  | | | m | | m | |  | m | m |  | |

For a default type conversion, your specification would take the following form:

*new\_columnname*:*new\_type* = [*explicit\_conversion\_function*] *old\_columnname* [Copy](javascript:void(0);)

For example, to produce an int8 column type:

int8col:int8 = uint64col[Copy](javascript:void(0);)

Where a manual conversion is required, your specification takes the form:

new\_columnname:new\_type = conversion\_function (old\_columnname)[Copy](javascript:void(0);)

For example:

day\_column:int8 = month\_day\_from\_date (date\_column)[Copy](javascript:void(0);)

The *new\_type* can be any of the destination types that are supported for conversions from the source (that is, any of the columns marked "m" in the above table). For example, you can use the conversion **hours\_from\_time** to convert a time to an int8, or to an int16, int32, dfloat, and so on. InfoSphere DataStage warns you when it is performing an implicit data type conversion, for example **hours\_from\_time** expects to convert a time to an int8, and will warn you if converting to a int16, int32, or dfloat.

The following table lists the available conversion functions. The source and destinations are always specified in terms of column names. Preliminary arguments are enclosed in square brackets, the source column name is enclosed in round brackets.

| **Conversion** | **Arguments** | **Output type** | **Description** | **Example** |
| --- | --- | --- | --- | --- |
| date\_from\_days\_ since | [*base\_date* (date)] (*number\_col* (int32)) | date | Converts an integer field into a date by adding the integer to the specified base date. The base\_date must be in the format *yyyy-mm-dd* and must be either double  quoted or a variable. | date\_col:date = date\_from\_days\_since ["1958-08-18"] (int\_col) |
| date\_from\_julian\_day | (*juliandate\_col* (uint32)) | date | Date from Julian day. | date\_col:date = date\_from\_julian\_day (julian\_col) |
| date\_from\_string | [*date\_format*] (*string\_col* (string)) | date | Converts the string to a date representation using the specified *date\_format*. By default the string format is *yyyy-mm-dd*. | date\_col:date = date\_from\_string ["%yyyy-%mm-%dd"] (string\_col) |
| date\_from\_timestamp | (*timestamp\_col*(timestamp) ) | date | Converts the timestamp to a date representation. | date\_col:date = date\_from\_timestamp (ts\_col) |
| date\_from\_ustring | [*date\_format*] (*string\_col* (ustring)) | date | Converts the string to a date representation using the specified *date\_format*. By default the string format is *yyyy-mm-dd*. | date\_col:date = date\_from\_ustring (ustring\_col, "%yyyy-%mm-%dd") |
| days\_since\_from\_date | [*source\_date* (date)] (*date\_col* (string)) | int32 | Returns a value corresponding to the number of days from *source\_date*to the specified date. *source\_date*must be in the form *yyyy-mm-dd* and can be quoted or unquoted. | dayssince\_col:int32 = days\_since\_from\_date ["1958-08-18"] (sourcedate\_col,) |
| decimal\_from\_decimal | [*r\_type*] (*source\_decimal\_col*(decimal)) | decimal | Decimal from decimal. | decimal\_col:decimal = decimal\_from\_decimal [ceil] (source\_col) |
| decimal\_from\_dfloat | [*r\_type*] (*source\_dfloat\_col*(dfloat)) | decimal | Decimal from dfloat. | decimal\_col:decimal = decimal\_from\_dfloat [ceil] (source\_col) |
| decimal\_from\_string | [*r\_type*] (*source\_string\_col* (string)) | decimal | Decimal from string. | decimal\_col:decimal = decimal\_from\_string [ceil] (source\_col) |
| decimal\_from\_ustring | [*r\_type*] (*source\_ustring\_col* (ustring)) | decimal | Decimal from ustring. | decimal\_col:decimal = decimal\_from\_ustring [ceil] (source\_col) |
| dfloat\_from\_decimal | [fix\_zero] (*source\_dec\_col* (decimal)) | dfloat | Dfloat from decimal. | dfloat\_col:dfloat = dfloat\_from\_decimal [fix\_zero] (source\_col) |
| hours\_from\_time | (*source\_time\_col* (time)) | int8 | Hours from time. | hours\_col:int8 = hours\_from\_time (time\_col) |
| int32\_from\_decimal | [*r\_type*, fix\_zero] (*source\_decimal\_col*(decimal)) | int32 | Int32 from decimal. | int32\_col:int32 = int32\_from\_decimal [ceil] [fix\_zero] (dec\_col) |
| int64\_from\_decimal | [*r\_type*, fix\_zero] (*source\_decimal\_col*(decimal)) | int64 | Int64 from decimal. | int64\_col:int64 = int64\_from\_decimal [ceil] (dec\_col) |
| julian\_day\_from\_date | (*date\_col* (date)) | uint32 | Julian day from date. | julianday\_col:uint32 = julian\_day\_from\_from\_date (date\_col) |
| lookup\_string\_from \_int16 | [*table\_definition* ], (*number\_col* (int16)) | string | Converts numeric values to strings by means of a lookup table. | gendercol:string = lookup\_string\_from\_int16 [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_ustring\_from \_int16 | [*table\_definition* ] (*number\_col* (int16)) | ustring | Converts numeric values to ustrings by means of a lookup table. | gendercol:ustring = lookup\_ustring\_from\_int16 [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_ustring\_from \_int32 | [*table\_definition* ] (*number\_col* (int32)) | ustring | Converts numeric values to ustrings by means of a lookup table.. | gendercol:ustring = lookup\_string\_from\_int32 [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_string\_from \_uint32 | [*table\_definition* ] (*number\_col* (uint32)) | string | Converts numeric values to strings by means of a lookup table. | gendercol:string = lookup\_string\_from\_uint16 [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_int16\_from \_string | [*table\_definition* ] (*string\_col* (string)) | int16 | Converts strings to numeric values by means of a lookup table. | int\_col:int16 = lookup\_int16\_from\_string [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_int16\_from \_ustring | [*table\_definition* ] (*ustring\_col* (ustring)) | int16 | Converts strings to numeric values by means of a lookup table. | int\_col:int16 = lookup\_int16\_from\_ustring [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_uint32\_from \_string | [*table\_definition* ] (*string\_col* (string)) | uint32 | Converts strings to numeric values by means of a lookup table. | int\_col:uint32 = lookup\_uint32\_from\_string [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lookup\_uint32\_from \_ustring | [*table\_definition* ] (*ustring\_col* (ustring)) | uint32 | Converts ustrings to numeric values by means of a lookup table. | int\_col:uint32 = lookup\_uint32\_from\_ustring [{default\_value = 2} ('f' = 1; 'm' = 2)] (gendercode) |
| lowercase\_string | (*instring\_col* (string)) | string | Convert strings to all lower case. Non-alphabetic characters are ignored in the conversion. | ostring\_col:string = lowercase\_string (istring\_col) |
| lowercase\_ustring | (*instring\_col* (ustring)) | string | Convert ustrings to all lower case. Non-alphabetic characters are ignored in the conversion. | ostring\_col:ustring = lowercase\_string (istring\_col) |
| mantissa\_from \_decimal | (*decimal\_col* (decimal)) | dfloat | Returns the mantissa from the given decimal | matissa\_col:dfloat = mantissa\_from\_decimal (dec\_col) |
| mantissa\_from\_dfloat | (*dfloat\_col* (dfloat)) | dfloat | Returns the mantissa from the given dfloat | matissa\_col:dfloat = mantissa\_from\_dfloat (dfloat\_col) |
| microseconds \_from\_time | (*time\_col* (time)) | int32 | Returns the microseconds from a time field. | msec\_col:int32 = microseconds\_from\_time (time\_col) |
| midnight\_seconds \_from\_time | (*time\_col* (time)) | dfloat | Returns the seconds-from-midnight from the supplied time. | midsec\_col:dfloat = midnight\_seconds\_from\_time (time\_col) |
| minutes\_from\_time | (*time\_col* (time)) | int8 | Returns the minutes from a time field. | minsec\_col:int8 = minutes\_from\_time (time\_col) |
| month\_day \_from\_date | (*date\_col* (date)) | int8 | Returns the day of month from a date field. | monthday\_col:int8 = month\_day\_from\_date (date\_col) |
| month\_from\_date | (*date\_col* (date)) | int8 | Returns the numeric month from a date field. | month\_col:int8 = month\_from\_date (date\_col) |
| next\_weekday\_from \_date | [*day*] (*date\_col* (date)) | date | Returns the date of the specified day of the week soonest after the source date (including the source date). *day* is a string specifying a day of the week. You can specify *day* by either the first three characters of the day name or the full day name. The *day* can be quoted in either single or double quotes or quotes can be omitted. | nextday\_col:date = next\_weekday\_from\_date [wed](date\_col) |
| notnull | (*any*) | int8 | Returns true (1) when an expression does not evaluate to the null value. | isnotnull\_col:int8 = notnull (test\_col) |
| null | (*any*) | int8 | Returns true (1) when an expression does evaluate to the null value | isnull\_col:int8 = null (test\_col) |
| previous\_weekday \_from\_date | [*day*] (*date\_col* (date)) | date | The destination contains the closest date for the specified day of the week earlier than the source date (including the source date). The *day* is a string specifying a day of the week. You can specify *day* by either the first three characters of the day name or the full day name. The *day* can be quoted in either single or double quotes or quotes can be omitted. | prevday\_col:date = previous\_weekday\_from\_date [wed](date\_col) |
| raw\_from\_string | (*string\_col* (string)) | raw | Returns a string in raw representation. | raw\_col:raw = raw\_from\_string (string\_col) |
| raw\_length | (*raw\_col* (raw)) | int32 | Returns the length of a raw field. | rawlength\_col:int32 = raw\_length (raw\_col) |
| seconds\_from\_time | (*time\_col* (time)) | dfloat | Returns the seconds from a time field. | sec\_col:dfloat = seconds\_from\_time (time\_col) |
| seconds\_since\_from \_timestamp | (*timestamp\_col*(timestamp)) | dfloat | Seconds since the time given by *timestamp.* | secsince\_col:dfloat = seconds\_since\_from\_timestamp (timestamp\_col) |
| string\_from\_date | [*date\_format*] (*date\_col*(date)) | string | Converts the date to a string representation using the specified *date\_format*. | datestring\_col:string = string\_from\_date [%dd-%mm-%yyyy] (date\_col) |
| string\_from\_decimal | [fix\_zero] (*decimal\_col*(decimal) | string | Returns a string from a decimal. | string\_col:string = string\_from\_decimal [fix\_zero] (dec\_col) |
| string\_from\_time | [*time\_format*] (*time\_col*(time)) | string | Converts the time to a string representation using the specified *time\_format*. The default time format is %hh:%nn:%ss. | timestring\_col:string = string\_from\_time [%hh:%nn:%ss.] (time\_col) |
| string\_from \_timestamp | [*timestamp\_forma*t] (*timestamp\_col*(timestamp)) |  | Converts the timestamp to a string representation using the specified *timestamp \_format*. The default timestamp format is %yyyy-%mm-%dd. %hh:%nn:%ss. | stringtimestamp\_col:string = string\_from\_timestamp [%yyyy-%mm-%dd. %hh:%nn:%ss.] (timestamp\_col) |
| string\_from\_ustring | (*string\_col* (ustring)) | string | Returns a string from a ustring. | string\_col:string = string\_from\_ustring (ustring\_col) |
| string\_length | (*string\_col* (string)) | int32 | Returns an int32 containing the length of a string. | length\_col:int32 = string\_length (string\_col) |
| substring | [*startPosition*,*len*] (*string\_col* (string)) | string | Converts long strings to shorter strings by string extraction. The *startPosition*specifies the starting location of the substring; *len*specifies the substring length. If *startPosition* is positive, it specifies the byte offset into the string from the beginning of the string. If *startPosition* is negative, it specifies the byte offset from the end of the string. | shorstring\_col:string = substring [5,10] (longstring\_col) |
| time\_from\_midnight \_seconds | (*dfloat\_col*(dfloat)) | time | Returns a time from aseconds-from-midnight field. | time\_col:time = time\_from\_midnight\_seconds (dfloat\_col) |
| time\_from\_string | [*time\_format*] (*string\_col* (string)) | time | Converts the string to a time representation using the specified *time\_format*. The default time format is %hh:%nn:%ss. | time\_col:time = time\_from\_string [%hh:%nn:%ss.] (string\_col) |
| time\_from\_timestamp | (*timestamp\_col*(timestamp)) | time | Time from timestamp. | time\_col:time = time\_from\_timestamp (timestamp\_col) |
| time\_from\_ustring | (*string\_col*(ustring)) | time | Returns a time from a ustring. | time\_col:time = time\_from\_ustring (string\_col) |
| timestamp\_from\_date | [*time*](*date\_col* (date)) | time stamp | Timestamp from date. The *time*argument optionally specifies the time to be used in building the timestamp result and must be in the form hh:nn:ss. If omitted, the time defaults to midnight. | timestamp\_col:timestamp = timestamp\_from\_date [08:20:33] (date\_col) |
| timestamp\_from \_seconds\_since | (*secondssince\_col*(dfloat)) | time stamp | Timestamp from a seconds since value. | timestamp\_col:timestamp = timestamp\_from\_seconds\_since (secondssince\_col) |
| timestamp\_from \_string | [*timestamp\_format*] (*string\_col* (string)) | time stamp | Converts the string to a timestamp representation using the specified *timestamp \_format*. By default, the string format is %yyyy-%mm-%dd hh:nn:ss. | timestamp\_col:timestamp = timestamp\_from\_string [%yyyy-%mm-%dd hh:nn:ss] (string\_col) |
| timestamp\_from\_time | [*date*](*time\_col* (time)) | time stamp | Timestamp from time. The *date*argument is required. It specifies the date portion of the timestamp and must be in the form *yyyy-mm-dd*. | timestamp\_col:timestamp = timestamp\_from\_time [1958-08-18] (time\_col) |
| timestamp\_from \_timet | (*timet\_col*(int32)) | time stamp | Timestamp from time\_t. The source field must contain a timestamp as defined by the UNIX time\_t representation. | timestamp\_col:timestamp = timestamp\_from\_timet (timet\_col) |
| timestamp\_from \_ustring | (*string\_col*(ustring)) | time stamp | Returns a timestamp from a ustring. | timestamp\_col:timestamp = timestamp\_from\_ustring (string\_col) |
| timet\_from \_timestamp | (*tstamp\_col*(timestamp)) | int32 | Time\_t from timestamp. The destination column contains a timestamp as defined by the UNIX time\_t representation. | timet\_col:int32 = timet\_from\_timestamp (tstamp\_col) |
| uint64\_from\_decimal | [*r\_type,* fix\_zero] (*dec\_col* (decimal)) | uint64 | Uint64 from decimal. | int\_col:uint64 = uint64\_from\_decimal [ceil, fix\_zero] (dec\_col) |
| uppercase\_string | (*string\_col*(string)) | string | Convert strings to all upper case. Non-alphabetic characters are ignored in the conversion. | string\_col:string = uppercase\_string (instring\_col) |
| uppercase\_ustring | (*string\_col*(ustring)) | ustring | Convert ustrings to all upper case. Non-alphabetic characters are ignored in the conversion. | ustring\_col:string = uppercase\_ustring (string\_col) |
| u\_raw\_from\_string | (*string\_col*(ustring)) | raw | Returns a raw from a ustring | raw\_col:raw = u\_raw\_from\_string (string\_col) |
| ustring\_from\_date | (*date\_col* (date)) | ustring | Returns a ustring from a date. | string\_col:ustring = ustring\_from\_date (date\_col) |
| ustring\_from\_decimal | (*dec\_col*(decimal)) | ustring | Returns a ustring from a decimal. | string\_col:ustring = ustring\_from\_decimal (dec\_col) |
| ustring\_from\_string | (*string\_col*(string)) | ustring | Returns a ustring from a string. | string\_col:ustring = ustring\_from\_string (string\_col) |
| ustring\_from\_time | (*time\_col* (time)) | ustring | Returns a ustring from a time. | string\_col:ustring = ustring\_from\_time (time\_col) |
| ustring\_from \_timestamp | (*timestamp\_col*(timestamp)) | ustring | Returns a ustring from a timestamp. | string\_col:ustring = ustring\_from\_timestamp (timestamp\_col) |
| ustring\_length | (*string\_col* (ustring)) | int32 | Returns the length of a ustring. | length\_col:int32 = ustring\_length (string\_col) |
| u\_substring | [*startPosition*,*len*] (*string\_col* (string)) | ustring | Converts long ustrings to shorter ustrings by string extraction. The *startPosition*specifies the starting location of the substring; *len*specifies the substring length. If *startPosition* is positive, it specifies the byte offset into the string from the beginning of the string. If *startPosition* is negative, it specifies the byte offset from the end of the string. | shorstring\_col:ustring = substring [5,10] (longstring\_col) |
| weekday\_from\_date | [*originDay*] (*date\_col*(date)) | int8 | Day of week from date. *originDay* is a string specifying the day considered to be day zero of the week. You can specify the day using either the first three characters of the day name or the full day name. If omitted, Sunday is defined as day zero. The *originDay*can be either single- or double-quoted or the quotes can be omitted. | dow\_int:int8 = [mon] (date\_col) |
| year\_day\_from\_date | (*date\_col*(date)) | int16 | Day of year from date (returned value 1-366). | doy\_col:int16 = year\_day\_from\_date (date\_col) |
| year\_from\_date | (*date\_col*(date)) | int16 | Year from date. | year\_col:int16 = year\_from\_date (date\_col) |
| year\_week\_from\_date | (*date\_col* (date)) | int8 | Week of year from date. | week\_col:int8 = year\_week\_from\_date (date\_col) |

*table\_definition* defines the rows of a string lookup table and has the following form:

{*propertyList*} ('*string*' = *value*; *'string'* = *value*; ... )[Copy](javascript:void(0);)

where:

* *propertyList* is one or more of the following options; the entire list is enclosed in braces and properties are separated by commas if there are more than one:
  + **case\_sensitive**. Perform a case-sensitive search for matching strings; the default is case-insensitive.
  + **default\_value =** ***defVal***. The default numeric value returned for a string that does not match any of the strings in the table.
  + **default\_string =** ***defString***. The default string returned for numeric values that do not match any numeric value in the table.
* *string* specifies a comma-separated list of strings associated with value; enclose each string in quotes.
* *value* specifies a comma-separated list of 16-bit integer values associated with *string*.

*date\_format* is the standard date formatting string described in [Date and time formats](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/dateandtimeformats.html?view=kc)

*R\_type* is a string representing the rounding type and should contain one of the following:

* **ceil**. Round the source field toward positive infinity. E.g, 1.4 -> 2, -1.6 -> -1.
* **floor**. Round the source field toward negative infinity. E.g, 1.6 -> 1, -1.4 -> -2.
* **round\_inf**. Round or truncate the source field toward the nearest representable value, breaking ties by rounding positive values toward positive infinity and negative values toward negative infinity. E.g, 1.4 -> 1, 1.5 -> 2, -1.4 -> -1, -1.5 -> -2.
* **trunc\_zero**. Discard any fractional digits to the right of the rightmost fractional digit supported in the destination, regardless of sign. For example, if the destination is an integer, all fractional digits are truncated. If the destination is another decimal with a smaller scale, round or truncate to the scale size of the destination decimal. E.g, 1.6 -> 1, -1.6 -> -1.

You can specify fix\_zero for decimal source columns so that columns containing all zeros (by default illegal) are treated as a valid decimal with a value of zero.

This tab allows you to specify options.

This tab allows you to specify the following:

* **Execution Mode**. The stage can execute in parallel mode or sequential mode. In parallel mode the input data is processed by the available nodes as specified in the Configuration file, and by any node constraints specified on the Advanced tab. In Sequential mode the entire data set is processed by the conductor node.
* **Combinability mode**. This is Auto by default, which allows InfoSphere® DataStage® to combine the operators that underlie parallel stages so that they run in the same process if it is sensible for this type of stage.
* **Preserve partitioning**. This is **Propagate** by default. If you have an input data set, it adopts **Set** or **Clear** from the previous stage. You can explicitly select **Set** or **Clear**. Select **Set** to request the next stage should attempt to maintain the partitioning.
* **Node pool and resource constraints**. Select this option to constrain parallel execution to the node pool or pools or resource pool or pools specified in the grid. The grid allows you to make choices from drop down lists populated from the Configuration file.
* **Node map constraint**. Select this option to constrain parallel execution to the nodes in a defined node map. You can define a node map by typing node numbers into the text box or by clicking the browse button to open the **Available Nodes** dialog box and selecting nodes from there. You are effectively defining a new node pool for this stage (in addition to any node pools defined in the Configuration file).

**Note**In the **Node map constraint** text box, you can enter jobs parameters as well as numbers. You can enter a single parameter, for example #testnode#, or you can enter a comma separated lists of parameters, for example #testnode#, #testnode2#. The browse button next to the text box will display a list of the node names from the last configuration file that was referenced by the job, but the browse button will not display the node names that were specified by the job parameters.

The Input page allows you to specify details about the incoming data set you are modifying. There is only one input link.

The Input page allows you to specify details about the incoming data set you are modifying. There is only one input link.

The General tab allows you to specify an optional description of the link. The Partitioning tab allows you to specify how incoming data on the source data set link is partitioned. The Columns tab specifies the column definitions of incoming data. The Advanced tab allows you to change the default buffering settings for the input link.

Details about Modify stage partitioning are given in the following section. See ["Stage Editors,"](https://www.ibm.com/support/knowledgecenter/SSZJPZ_11.7.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Stage_Editors.html?view=kc) for a general description of the other tabs.

The Partitioning tab allows you to specify details about how the incoming data is partitioned or collected before the modify is performed.

By default the stage uses the auto partitioning method.

If the Modify stage is operating in sequential mode, it will first collect the data before writing it to the file using the default auto collection method.

The Partitioning tab allows you to override this default behavior. The exact operation of this tab depends on:

* Whether the Modify stage is set to execute in parallel or sequential mode.
* Whether the preceding stage in the job is set to execute in parallel or sequential mode.

If the Modify stage is set to execute in parallel, then you can set a partitioning method by selecting from the **Partition type**drop-down list. This will override any current partitioning.

If the Modify stage is set to execute in sequential mode, but the preceding stage is executing in parallel, then you can set a collection method from the **Collector type** drop-down list. This will override the default auto collection method.

The following partitioning methods are available:

* **(Auto)**. InfoSphere® DataStage® attempts to work out the best partitioning method depending on execution modes of current and preceding stages and how many nodes are specified in the Configuration file. This is the default method for the Modify stage.
* **Entire**. Each file written to receives the entire data set.
* **Hash**. The records are hashed into partitions based on the value of a key column or columns selected from the **Available**list.
* **Modulus**. The records are partitioned using a modulus function on the key column selected from the **Available** list. This is commonly used to partition on tag fields.
* **Random**. The records are partitioned randomly, based on the output of a random number generator.
* **Round Robin**. The records are partitioned on a round robin basis as they enter the stage.
* **Same**. Preserves the partitioning already in place.
* **Db2®**. Replicates the Db2 partitioning method of a specific Db2 table. Requires extra properties to be set. Access these properties by clicking the properties button.
* **Range**. Divides a data set into approximately equal size partitions based on one or more partitioning keys. Range partitioning is often a preprocessing step to performing a total sort on a data set. Requires extra properties to be set. Access these properties by clicking the properties button.

The following Collection methods are available:

* **(Auto)**. This is the default collection method for the Modify stage. Normally, when you are using Auto mode, InfoSphere DataStage will eagerly read any row from any input partition as it becomes available.
* **Ordered**. Reads all records from the first partition, then all records from the second partition, and so on.
* **Round Robin**. Reads a record from the first input partition, then from the second partition, and so on. After reaching the last partition, the operation starts over.
* **Sort Merge**. Reads records in an order based on one or more columns of the record. This requires you to select a collecting key column from the **Available** list.

The Partitioning tab also allows you to specify that data arriving on the input link should be sorted before the modify operation is performed. The sort is always carried out within data partitions. If the stage is partitioning incoming data the sort occurs after the partitioning. If the stage is collecting data, the sort occurs before the collection. The availability of sorting depends on the partitioning or collecting method chosen (it is not available for the default auto methods).

Select the check boxes as follows:

* **Perform Sort**. Select this to specify that data coming in on the link should be sorted. Select the column or columns to sort on from the **Available** list.
* **Stable**. Select this if you want to preserve previously sorted data sets. This is the default.
* **Unique**. Select this to specify that, if multiple records have identical sorting key values, only one record is retained. If stable sort is also set, the first record is retained.

If NLS is enabled an additional button opens a dialog box allowing you to select a locale specifying the collate convention for the sort.

You can also specify sort direction, case sensitivity, whether sorted as ASCII or EBCDIC, and whether null columns will appear first or last for each column. Where you are using a keyed partitioning method, you can also specify whether the column is used as a key for sorting, for partitioning, or for both. Select the column in the **Selected** list and right-click to invoke the shortcut menu.