Transformation Mapping Validity

Introduction

Transformation mappings must be valid for a transformation to be valid. This page covers how to identify invalid mappings and view the validation errors associated with them, as well as how to resolve validation errors.

Validation Errors

Transformations are treated the same as other project components when identifying component validity on a project scale. That is, you can identify invalid transformations on the design canvas and in the project pane as covered in Component Validity. Validation errors within a mapping are included in the validation of transformations.

In order for a transformation to be valid, it must not have any validation errors within the transformation itself. To be considered valid, a transformation must meet these rules:

- A mapping cannot contain references to non-existent fields or variables.
- A mapping cannot contain data type conflicts.
- A mapping must use valid script syntax.
- A target loop node cannot have multiple sources.
- A schema must be provided for an adjacent source or target activity.

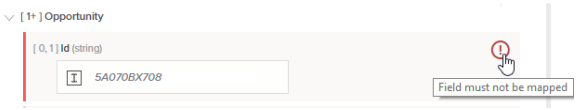
In addition, certain target fields may require a mapping, or may not allow a mapping. Additional details are provided under Validation Rules later on this page.

To resolve those errors, you can identify errors within the transformation itself and make adjustments accordingly. Invalid mappings are displayed either for a entire target field block, or for individual mapped objects within a target field block.

Entire Target Field

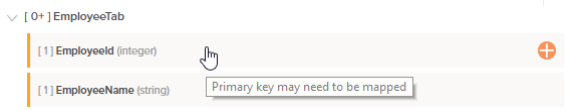
When an entire target field mapping is invalid, a red vertical line appears along the left of the target field block, an exclamation mark is displayed in the top right of the target field. Upon hover over the exclamation mark, the reason for the mapping being invalid is listed.

This type of invalid mapping may result from mapping of fields that cannot be mapped, for example by attempting to map to a unique object ID on a Salesforce insert activity used as a target.



When an entire target field mapping is potentially invalid, an orange vertical line appears along the left of the target field block. Upon hover over the field, the reason for the mapping potentially being invalid is listed.

This type of invalid mapping may result from the absence of a mapping that may be required, for example from a primary key not being mapped on a database insert activity used as a target.



Invalid or potentially invalid mappings may also result from multiple objects (such as more than one source object or variable) being mapped to a target field without any logic specifying how to process those objects.

To resolve invalid mappings, either remove the mapping, or open the field in script mode to add processing logic. Potentially invalid mappings are intended to flag a potential problem but can be left "as is" if you have determined the mapping is valid.

To remove a mapping, click the trash icon on the mapped object.

To open a field with an invalid mapping in script mode to add processing logic, click the exclamation mark to show the script icon and menu options. For a field with a potentially invalid mapping, the script icon is already present. Click **Add Script** to enter script mode.

On This Page

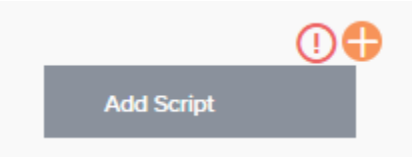
Related Articles

- Component Validity
- Database Insert Activity
- Design Canvas
- Nodes and Fields
- Project Pane
- Salesforce Insert or Update Activity

Related Topics

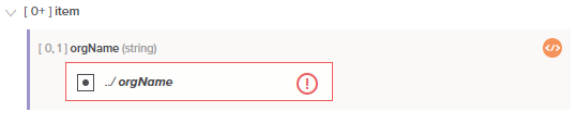
- Cloud Studio
- Connectors
- Schemas
- Scripts
- Transformation Mapping
- Transformations
- Variables

Last updated: Aug 09, 2019

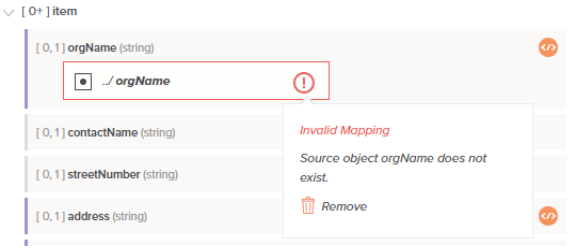


Individual Mapped Object

When an individual mapped object within a target field is invalid, the mapped object is outlined in red with an exclamation mark displayed on the right:



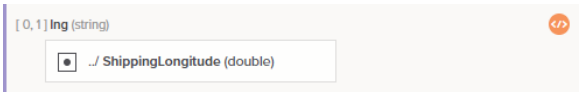
Click the exclamation mark to display the reason for the mapping being invalid. To remove the mapping, click the trash icon or **Remove**.



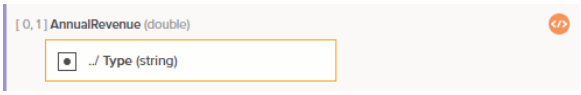
Individual mappings may be invalid if a mapped source object is no longer present after editing a schema, if there is a required/optional mismatch with the cardinality of a target field, or if source and target fields have incompatible data types.

The mapping of source and target fields with potentially incompatible data types is flagged for all schemas regardless of if they have defined data types, such as those defined from a Salesforce or database activity, or if they have data type labels only, such as those defined from a sample file or that have been manually defined (for these types of schemas, all fields are handled as strings regardless of whether they have data type labels).

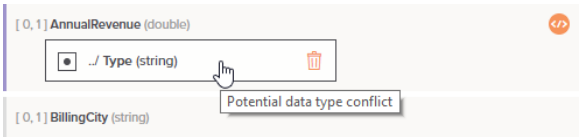
A data type mismatch does not necessarily indicate there is a problem with the mapping. For example if a source field with an integer data type is mapped to a target field with a string data type, the resulting data will be a string that contains a numerical value. Source fields with any data type can be mapped to a target field with a string data type, so these mappings are not flagged in any way.



However, if the reverse is true, and a source field with a string or other data type is mapped to a mismatched target data type, such mappings may be flagged as potentially incompatible. Mapped fields with data types that are potentially incompatible are displayed with an orange border:



Upon hover, the reason for the potential incompatibility is listed. To remove the mapping, click the trash icon.



Note that the indication of a potential data type mismatch appears only for simple mappings of a source object to a target field, and is not displayed if the target field contains scripting logic.

For target fields that are mapped using scripting logic, after the script is processed and the resulting output is evaluated, the value will be converted to the data type of the target field it is being mapped to. The user is expected to know, based on the output being converted to the target field data type, if this final output is the desired result.

As an example, let's say the target field is a `double` data type. Regardless of the type of data in the script, after the script is processed, the final output for the mapping will be converted to a `double`. This is illustrated with the transformation script examples below.

Transformation Script 1

```
<trans>
1 + "string"
</trans>
```

The script above evaluates to a value of `"1string"` with a `string` data type. Then, during processing of the transformation mapping, this output is converted to a `double` data type. Because the output begins with a digit, any beginning digit(s) would carry over to the final mapping output, as numbers are valid for a `double`. The final output for the mapping is a value of `1` with a `double` data type.

Transformation Script 2

```
<trans>
"string" + 1
</trans>
```

The second script, above, evaluates to a value of `"string1"` with a `string` data type. During processing of the transformation mapping, this output is converted to a `double` data type. In this case, because a non-digit value, which is not valid for a `double`, is at the beginning of the returned value, no value is carried over. Instead, the target field takes the default value for a number; that is, the final output for the mapping is a value of `0` with a `double` data type.

In some cases, you may wish to override the data type of the target field and actually carry over output using a different data type. For these, you can use the `Format()` or `FormatDate()` functions within the target field mapping script. Make sure that the function is in the last statement of the script, as the returned value of the script always comes from the final line.

Validation Rules

In order for a transformation to be valid, it must be configured properly. The error messages covered below are displayed when viewing the validation errors associated with the transformation as a component from the design canvas or project pane as covered in Component Validity.

If a transformation has not been configured or is misconfigured, this validation error messages will be returned:

> *Transformation is not configured properly.*

This message most commonly appears when you have added a new transformation to an operation and it has not been configured yet. To resolve, open the transformation configuration screen, then configure the transformation accordingly.

In addition, in order for a transformation to be valid, it must not have any validation errors within the transformation itself. To be considered valid, a transformation must meet these rules:

- A mapping cannot contain references to non-existent fields or variables.
- A mapping cannot contain data type conflicts.
- A mapping must use valid script syntax.
- A target loop node cannot have multiple sources.
- A schema must be provided for an adjacent source or target activity.

In addition, certain target fields may require a mapping, or may not allow a mapping. Invalid mappings are visually indicated within the transformation configuration screen, as covered under Validation Errors earlier on this page.

Depending on the error, the appropriate variation of these possible error messages will be returned if this rule is not met:

> *Mapping refers to a non-existent [source / target / variable] field $[path].*
>
> *Potential data type conflict in mapping.*
>
> *Target field $[node.name] [must be mapped / cannot be mapped].*
>
> *records: Error on line [#]: [Script validation syntax error].*
>
> *Mappings of a target loop node depend upon more than one source loop node.*
>
> *[Source / Target] schema must be provided.*

To resolve, try these troubleshooting tips:

- If you have references to non-existent fields, data type conflicts, or other invalid mappings, either find the invalid mapping and unmap it or check the file schema to make sure all fields are accounted for and have compatible data types. If you have references to non-existent variables, check to make sure the variable exists.
- If you have script validation errors, check the mapping and make adjustments within the script. Validation messages are also displayed below each script reported on a line-by-line basis. That is, after resolving an error on one line, additional syntax errors to resolve may be reported for subsequent lines. Continue making changes until the message indicates the script is valid.
- If you have a target loop node that depends on more than one source loop node, follow the instructions provided in Mapping from a Multi-Instance Source to Single-Instance Target under Nodes and Fields.
- If you have source or target activities adjacent to the transformation, make sure you provide a file schema for each one. File schemas can be provided from within the activity itself during its configuration (see documentation for each connector), or by defining a file schema from directly within the transformation.

In addition, if a transformation is invalid for some other reason that cannot be readily determined, this error message will be returned:

> *Transformation is invalid.*