



Rocket Uniface Library 10.4

structToComponent

Converts data from a Struct or collection of Structs into the entities and occurrences in the component.

`structToComponent {/firetriggers} StructSource`

Example: `structToComponent vStruct`

Qualifiers

Table: Qualifiers

Qualifier	Description
/firetriggers	Causes the preDeserialize and postDeserialize triggers to be fired. These triggers can be used to provide additional processing, for example when preparing data to be loaded and reconnected into a component that contains data.

Parameters

Table: Parameters

Parameter	Data Type	Description
StructSource	Struct	Variable, parameter, or non-database field referring to the source Struct; must be of type struct or type any

Return Values

Table: Values Commonly Returned in \$status after structToComponent

Value	Meaning
<0	An error occurred. \$procerror contains the exact error. \$procerrorcontext contains the details.
0	Struct successfully created. However, non-fatal errors may occur during conversion, because everything that is not recognized or usable is ignored. Warnings about such conditions are made available in \$procReturnContext . See \$procReturnContext for structToComponent .

Common Errors

Table: Values Commonly Returned by \$procerror

Error Number	Error Constant	Meaning
-1905	STRUCTERR_INPUT	Input struct data is not valid. For example, the struct variable may have been declared, but not initialized.

Use

Allowed in all component types.

Description

`structToComponent` transfers data from a Struct or collection of Structs into a component. The data is loaded directly into the component's data structure.



Note: `structToComponent` does not interpret or initiate data validation, so the data loaded by `structToComponent` can include duplicates of occurrences already available in the component.

Annotations

When preparing a Struct programmatically for conversion to a component, you can use `$tags` to set the `u_type` annotation tag. Using this tag can help to resolve ambiguity if a field and an entity have the same name. Matching the Struct against the component is name based, so the `u_type` tag may not be required—it is clear from the component structure that one thing is a field and another thing is an entity.

Table: Annotation Tags for Uniface Component-Struct Conversions

Tag	Allowed Values	Comments
<code>u_type</code>	<code>component</code> <code>entity</code> <code>occurrence</code> <code>field</code>	Each node in a component Struct has a <code>u_type</code> annotation that indicates the object type.
For nodes that have the tag <code>u_type="occurrence"</code> , the following tags are also supported. These can be used if you are using the Struct to manipulate data prior to a reconnecting the data to its source. For more information, see Metadata for Reconnect .		
<code>u_id</code>	<i>OccID</i>	Uniface-generated occurrence identifier
<code>u_crc</code>	<i>Checksum</i>	CRC checksum of the occurrence
<code>u_status</code>	<code>est</code> (exists in DB) <code>mod</code> (modified) <code>new</code> (new) <code>del</code> (delete)	Modification status of the occurrence.

Reconnect Loaded Data

If you are using Struct to transport disconnected record sets, the Struct may contain occurrence processing tags. These tags will be available if the Struct was created with `componentToStruct/reconnecttags`. In this case, the `structToComponent` statement sets state flags used by `reconnect`.

You should use a `reconnect` statement immediately after the `structToComponent` command. `reconnect` removes duplicates of occurrences, removes occurrences marked for deletion from the component, and sets the appropriate modification flags. For more information, see [Reconnect Process](#).

Note: The `structToComponent` statement only handles the reconnect processing tags if the `u_type` is present and has a value of occurrence.

Triggers

The `structToComponent` statement fires the following triggers that can be used to customize how the Struct is loaded into a component:

- [trigger preDeserialize](#)—fired immediately before an occurrence is loaded into a component. The new occurrence is not yet available and cannot be accessed.
- [trigger postDeserialize](#)—fired immediately after an occurrence is loaded into a component. The new occurrence is available and can be accessed. For example, use this trigger if an occurrence can be discarded, or the value for a derived field can be calculated.

Name Matching Rules

The `structToComponent` is primarily name driven. The following name matching rules apply:

- Tag names (annotations) are case sensitive. Tags should only occur once, else the value is empty.
- All tag value matching against component structure elements is case insensitive.
- Entity names or field names can be non-qualified (ENTITYNAME, or FIELDNAME), partially qualified (FIELDNAME.ENTNAME), or fully qualified (ENTITY.MODEL for an entity or FIELD.ENTITY.MODEL for a field). If an entity is not fully qualified matching it to a component structure can be ambiguous, in which case the outcome is not defined.

This behavior is the same as statements such as `retrieve/e Entity`, where the specified name matches two different entities, each within a different model. Field names can normally be unqualified, as they are implicitly qualified by the location within the context of an entity.

In case of ambiguity (identical names of a field and an entity drawn on the same level), names must be fully qualified, or tags can be used to specify the type of the object.

Conversion Logic

The top-most Struct to be converted must represent a component or an entity.


Note: If an entity is the starting point, it is not necessarily the top-level entity. It can be an entity that is nested inside the component structure.

From the starting point, the `structToComponent` conversion routine processes each member Struct in turn. For each Struct:

- If no tag is specified, it matches the Struct to the component structure based on the name.
- If no name match is found, a warning (STRUCTERR_NO_MATCHING_NAME) with related information is put in `$procReturnContext`, and the Struct (and all its children) is skipped.
- If the `u_type` tag is specified, the match is based on this value, and the name.

More specifically, the conversion routine does the following:

1. Finds the starting point.
 - a. Check whether the top-level Struct (or Structs, in a collection) has a `u_type` tag. If it does, the value must be `component` or `entity`; otherwise a warning is put in `$procReturnContext`.
 - b. Match the Struct name to a component or entity name.
 - If a match is found with the component name, continue at step 2.
 - If a match is found with an entity name, continue at step 3.
 - If no match is found, raise a warning in `$procReturnContext`.
2. Processes the component Struct. For each Struct member:
 - a. Check whether the member has a `u_type` tag. If it does, the value must be `entity`; otherwise a warning is put in `$procReturnContext`.
 - b. Match the name of the Struct member to top-level entity in the current component structure.
 - If a match is found, continue at step 3.
 - If no match is found, raise a warning in `$procReturnContext`.
3. Processes each entity Struct. For each Struct member:
 - a. Check whether the member has a `u_type` tag. If it does, the value must be `occurrence`; otherwise a warning is put in `$procReturnContext`.
 - b. For each occurrence Struct found, continue at step 4.

 **Note:** The Struct name is not used for matching.

4. Processes each occurrence Struct member.
 - a. Create a new occurrence at the end of the list of existing occurrences in the current entity.
 - b. For each member, check whether it has a `u_type` tag. If it does, the value must be `entity` or `field`; otherwise a warning is put in `$procReturnContext`.
 - c. Match the name of the member Struct with an entity or field. If `u_type` is not specified, it first tries to match an entity, and then a field.

If no match is found, raise a warning in `$procReturnContext`
5. Processes each field Struct member, assigning the value specified by the Struct to the field, and converting the data type if required.

\$procReturnContext for structToComponent

`$procReturnContext` contains context and error information about the conversion, in the form of a Uniface list.

```
Context=structToXml ;}
{Infos=Number ;
{Warnings=Number ;}
{Errors=Number ;}
{DETAILS=ID=MsgNum !!;SEVERITY=Type !!;MNEM=Mnemonic
!!;DESCRIPTION=ErrorDescription!!;CURRENTSTRUCT=Struct !!;ADDITIONAL=SPECIFIEDNAME=StructName
!!!;{INDEX=N !!!;}
EXPECTEDTYPE=ExpectedValue} { !!!;TAGNAME=u_type !!!;TAGVALUE=ValueID= ...}
```

Table: Items Returned by \$ProcReturnContext

Item	Description
Context = <i>Context</i>	Value indicating the previously executed command that set \$procReturnContext:structToComponent
Detail = <i>String</i>	Messages, warnings, and non-fatal errors encountered during processing, and additional information, structured as a list.

Table: Items Returned in Detail of \$procReturnContext

Item	Description
ID	Message number
MESSAGE	Message text
SEVERITY	Importance of the issue; one of INFO, WARNING, or ERROR
MNEM	Mnemonic for the specified (numeric) ID. One of: <ul style="list-style-type: none">• USTRUCTERR_NO_MATCHING_NAME—Struct name cannot be matched with an object name according to the matching rules• UTAGVALUE_NOT_APPLICABLE—Struct has a u_type tag , but its value does not match the component structure.• UTAGVALUE_NOT_RECOGNIZED—Struct has an invalid value, for example, because the wrong case is used. Values are case-sensitive.
DESCRIPTION	Short description of the issue.
CURRENTSTRUCT	List of all preceding parents, starting from the top. Each parent is described by its name (which can be empty) and index number. The top-level parent has no index number.
NAME	Name of the current Struct or Struct member
INDEX	Index number of the current Struct in the current
ADDITIONAL	Uniface sublist of additional information about the Struct (member) causing the message. This information is provided if there is more detailed information to report, such as unexpected tags or tag values.
TAGNAME	Name of the annotation, if specified. When converting to components, the only allowed tag name is u_type
TAGVALUE	Value of u_type.
EXPECTEDTYPE	Expected component object type for the context, or as specified by the u_type tag

Example: Information Returned in \$procReturnContext

The following shows the type of information returned in **\$procReturnContext** for **structToComponent** (formatted for readability):

```
Context=StructToComponent;  
Warnings=2;  
DETAILS=
```

```

ID=-1161!!;
SEVERITY=Warning!!;
MNEM=<STRUCTERR_NO_MATCHING_NAME>!!;
DESCRIPTION=No matching name found during conversion from struct!!;
CURRENTSTRUCT=ORDER->OCC{1}->ORDER_I{1}!!;
ADDITIONAL=
SPECIFIEDNAME=ORDER_I!!!;
EXPECTEDTYPE=entity or field!;
ID=-1160!!;
SEVERITY=Warning!!;
MNEM=<STRUCTERR_TAGVALUE_NOT_APPLICABLE>!!;
DESCRIPTION=Struct tag value not applicable in conversion from struct!!;
CURRENTSTRUCT=ORDER->OCC{1}->SHIP_TO{1}!!;
ADDITIONAL=
TAGNAME=u_type!!!;
TAGVALUE=component!!!;
EXPECTEDTYPE=entity or field

```

Example: Creating and Converting a Struct to a Component Structure

The following very simple code example:

- Creates a Struct for an ORDER entity containing an occurrence with two fields: ORDER_ID and SHIP_TO
- Displays the Struct in an OUTPUT field
- Transfers the Struct to the component's data structure

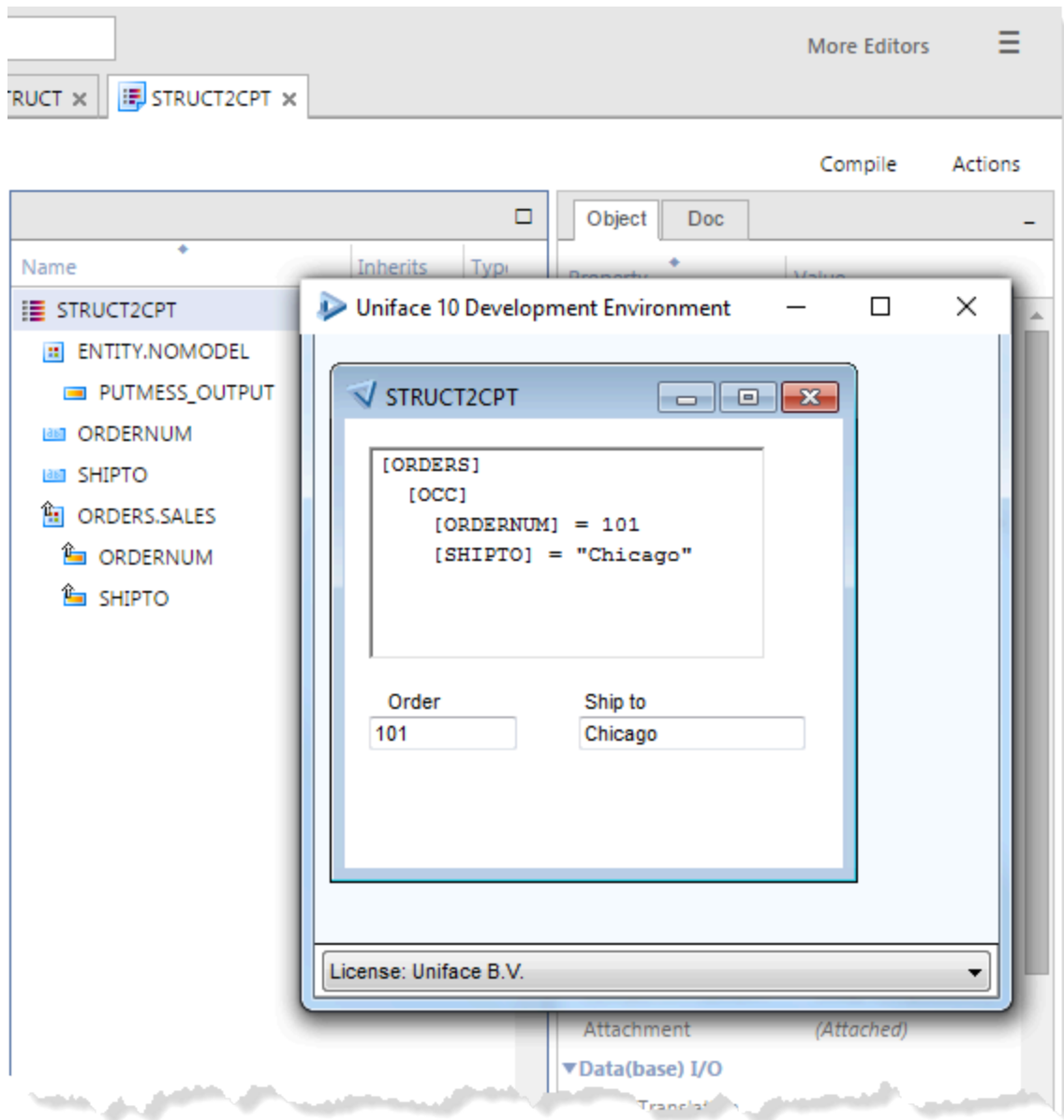
```

function createStruct
variables
    struct vStruct, vFld
endvariables
; Create a Struct with field names and values
vFld->ORDERNUM = 101
vFld->SHIPTO = "Chicago"
; Create a Struct named ORDERS
vStruct->$name = "ORDERS"
; Assign the Field Struct to a new member called OCC, of the ORDER Struct
vStruct->OCC = vFld
PUTMESS_OUTPUT = vStruct->$dbgstring
structToComponent vStruct
end

```

The following illustration shows the resulting form (in test mode, with the Component Editor showing the component structure).

Figure: Struct2Comp Test Form



Related concepts

[Transforming Complex Data Using Structs](#)
[componentToStruct](#)
[\\$procReturnContext](#)

Related tasks

[Structs for Uniface Component Data](#)