Rocket Uniface Library 10.4

Notices

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters

77 4th Avenue, Suite 100

Waltham, MA 02451-1468

USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country and Toll-free telephone number

- United States: 1-855-577-4323
- Australia: 1-800-823-405
- Belgium: 0800-266-65
- Canada: 1-855-577-4323
- China: 400-120-9242
- France: 08-05-08-05-62
- Germany: 0800-180-0882
- Italy: 800-878-295
- Japan: 0800-170-5464
- Netherlands: 0-800-022-2961
- New Zealand: 0800-003210
- South Africa: 0-800-980-818
- United Kingdom: 0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support. In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

Table of contents

ProcScript: Data Types

Data types define the characteristics of the data and determine how Uniface handles the data during processing and storage. They are specified for parameters, variables, and return values in ProcScript, as well as for fields, global variables, and component variables.

For more information, refer to the Uniface Data Types topic.

Related concepts

- **Uniface Data Types**
- **Transforming Complex Data Using Structs**
- **Storage Formats**

Related reference

- **Data Handling in ProcScript**
- **Packing Codes**

any

Uniface data type for any kind of data. It can used as a parameter/variable data type in all types of modules.

Use

Use in the `params` blocks of functions and the `variables` block for component variables.

To define an `any` global variable, choose `$` in the Data Type list of the editor.

Description

A variable or parameter with this data type can be assigned a value of any other data type, at which point it takes on the data type of its value.

boolean

Uniface data type that holds a Boolean value of `0` or `1`.

`boolean` *ParameterName* `:` *Direction*

`boolean` *VariableName*

Description

The Boolean data type is interpreted as either `TRUE` or `FALSE`. An empty value, and the values `0`, `F`, `f`, `N`, and `n` are interpreted as `FALSE`. All other values are interpreted as `TRUE`.

```
params
 boolean pCreditApproved : INOUT
```

```
endparams
```

Related concepts

- Data Types
- Packing Codes for Boolean Data

Related reference

- Data Handling in ProcScript

date

Uniface data type that holds a sequence of numbers representing a date, in the format *ccyymmdd*.

`date` *ParameterName* `:` *Direction*

`date` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

The `$date` ProcScript function can be used to explicitly convert data to the `date` data type.

Related concepts

- $date
- Date and Time Data
- Packing Codes for Date and Time Data

Related reference

- Data Handling in ProcScript

datetime

Uniface data type that holds a sequence of numbers representing a date and time, in the format *ccyymmddhhnnsstt*.

`datetime` *ParameterName* `:` *Direction*

`datetime` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

For the Datetime data type, the following defaults apply for partial values:

- If both the date and time have been omitted, the value is assumed to be NULL.
- If the date has been omitted while a time is specified, the day, month and year default to the current day, the current month and the current year, respectively.
- If an incomplete date is specified, a missing day defaults to 1, a missing month defaults to the current month, and a missing year defaults to the current year.

The `$datim` ProcScript function can be used to convert data to `datetime`.

Related concepts

$datim
Date and Time Data
Packing Codes for Date and Time Data

Related reference

Data Handling in ProcScript

entity

Uniface data type for transferring occurrences of entities between components.

```
params
```

`entity` *EntityName*{`.` *ModelName*} `:` *Direction*

```
endparams
```

Use

Use in `params` blocks to define the data type of a parameter.

Entity parameters cannot be used in entries and functions (global or local ProcScript modules).

Description

An entity parameter transfers all occurrences of the specified entity from one component to the other component.

> **Note:** Only fields whose **Is External** property is `T` can be included in the entity parameter.

For more information, see Entity and Occurrence Parameters.

Related concepts

params...endparams
Modeled Entities

Related reference

Is External

float

Uniface data type that holds a number with a floating decimal point.

`float` *ParameterName* `:` *Direction*

`float` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

Uniface internally supports a platform-independent precision of 38 digits for the mantissa, plus one for rounding, and four digits for the exponent, as follows:

```
+/- (99,999,999,999,999,999,999,999,999,999,999,999,999e+/- 9999)
```

When storing data, it depends on the DBMS whether this high precision is supported.

Fields defined as data type `float` or `numeric` can accept a scientific notation. For example, you could enter `1.23e-5` in a field with data type `float` and `123e-5` in a field with data type `numeric`.

If no display format is specified for a field with data type float, the field syntax length is used when formatting the value.

Related concepts

- **Extracting Values From Numeric Data**
- **Packing Codes for Numeric Data**
- **Floating Point Packing Codes (F)**

handle

Uniface data type containing a single reference to a component instance, entity, occurrence, field, or OCX object.

In a `params` block:

`handle` *ParameterName* `:` *Direction*

In a `variables` block:

{`public` | `partner`} `handle` {*VariableName*}

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining non-database fields, global variables, and component variables.

Description

Handles can be designated as `public` or `partner`. Public handles can only access public operations. Partner handles can access partner operations and public operations.

> **Note:** Do not convert `handle` to string and back.

`handle`

In the following example, the handle of a Uniface component is passed as a parameter to the exec operation of another component and assigned to a component variable. The handle in the component variable is used to activate operations on the component.

```
operation exec
 params
 handle pWriter : IN
 endparams
 $hWriter$ = pWriter
 $hWriter$->startdocument("true")
end; exec
```

Related concepts

- Handles

image

Uniface data type containing binary data.

`image` *ParameterName* `:` *Direction*

`image` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

Use the `image` data type for fields that contain images from the database, glyphs, disk files, or third-party filters.

No arithmetic operations can be applied to this kind of data.

It is not possible to directly compare two parameters, variables, or fields that use the data type `raw` or `image`—they will always evaluate to `True`. An alternative is to calculate and compare a hash of the data.

Related concepts

- Image Handling
- Packing Codes for Other Data

Related reference

Data Handling in ProcScript

lineardate

Uniface data type that holds an integer representing a number of days.

`lineardate` *ParamName* `:` *Direction*

`lineardate` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

A `lineardate` can have a value from `0` (days) through `3652425` (10,000 years, that is, 3,652,425 days). Only a whole number of days is allowed; that is, you cannot specify hours, minutes and so on.

```
params
 date pContractDate : IN
 date pExecutionDate : IN
 lineardate pWaitPeriod : OUT
endparams
pWaitPeriod = pExecutionDate - pContractDate
```

Related concepts

Date and Time Data
Packing Codes for Date and Time Data

Related reference

Data Handling in ProcScript

lineardatetime

Uniface data type that represents a number of days, including fractions of days.

`lineardatetime` *ParamName* `:` *Direction*

`lineardatetime` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

Linear Date and Time represents a number of days. Partial days (hours, minutes, seconds) can be expressed as a

fraction of a day.

Related concepts

- Date and Time Data
- Packing Codes for Date and Time Data

Related reference

- Data Handling in ProcScript

lineartime

Uniface data type that holds an integer that represents a number of hours.

`lineartime` *ParamName* `:` *Direction*

`lineartime` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

A `lineartime` can have a value from `0` to `24`. If more specific time information is required, use the `lineardatetime` or `lineardate` data type format.

```
params
 time pTimeStamp : IN
 lineartime pElapsedTime : OUT
endparams
pElapsedTime = $clock-pTimeStamp
```

Related concepts

- Date and Time Data
- Packing Codes for Date and Time Data

Related reference

- Data Handling in ProcScript

numeric

Uniface data type specifying a number to a maximum of 38 decimal places, including `+`, `-`, and decimal point `.`

`numeric` *ParamName* `:` *Direction*

`numeric` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of the specified parameter or variable. The data type can also be selected when defining fields, global variables, and component variables.

Description

The `numeric` data type supports arithmetic functions, fractions (scaling), decimal points, and positive and negative values. The number of decimals stored depends on the packing code.

Fields defined as data type `float` or `numeric` can accept a scientific notation. For example, you could enter `1.23e-5` in a field with data type `float` and `123e-5` in a field with data type `numeric`.

Uniface arithmetic expressions use string representations of the numbers as input, and return a string representation of the result, truncated at 38 digits. If the 39th digit is 5 or higher, the number is rounded upwards.

```
params
 numeric pPrice, pQuantity, pTotal : IN
endparams
pTotal = pPrice * pQuantity
```

Related concepts

- Extracting Values From Numeric Data
- Packing Codes for Numeric Data
- Numeric Strings (C20)
- Numeric Constant
- $encode
- $decode

Related reference

- Data Handling in ProcScript
- $random

occurrence

Uniface data type containing a single occurrence of an entity. For use in parameters.

`params`

`occurrence` *EntityName*{`.` *ModelName*} `:` *Direction*

`endparams`

Use

Use in `params` blocks of `operation` definitions to define the data type of a parameter.

Occurrence parameters cannot be used in entries and functions (global or local ProcScript modules) and in global ProcScript.

Description

An occurrence parameter transfers the current occurrence of the specified entity from one component to another component.

> **Note:** Only fields whose **Is External** property is `T` can be included in the occurrence parameter.

Related concepts

- Occurrences
- params...endparams

Related reference

- Is External

raw

Uniface data type that holds binary data.

`raw` *ParameterName* `:` *Direction*

`raw` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

Use the `raw` data type for fields containing raw data that should not be converted or processed, such as pictures and sound.

`raw` data is encoded as `UTF-8`.

No arithmetic operations can be applied to this kind of data.

It is not possible to directly compare two parameters, variables, or fields that use the data type `raw` or `image`—they will always evaluate to `True`. An alternative is to calculate and compare a hash of the data.

Raw data is not supported in Service Stored Procedures (SSP) for most databases. It is supported for SSP only on Sybase.

Comparing `raw` or `image` data

You can calculate a hash of the data and compare the hash instead of the content. The following code checks whether two files are the same.

```
variables
 string vFile1Name, vFile2Name, vFile1MD5Hash, vFile2MD5Hash
 raw vFile1Raw, vFile2Raw
```



```
 numeric vFile1Size, vFile2Size
endvariables
; Create a hash of
lfileload/raw vFile1Name, vFile1Raw
vFile1Size = $status
vFile1MD5Hash = $encode("HEX", $encode("MD5", vFile1Raw))
lfileload/raw vFileName2, vFile2Raw
vFile2Size = $status
vFile2MD5Hash = $encode("HEX", $encode("MD5", vFile2Raw))
if (vFile1Size = vFile2Size & vFile1MD5Hash = vFile2MD5Hash)
 message/info "[%%vFile1Name] and [%%vFile2Name] are identical."
else
 message/error "[%%vFile1Name] and [%%vFile2Name] are different!"
endif
```

Related concepts

- Data Types
- Packing Codes for Other Data

Related reference

- Data Handling in ProcScript

string

Uniface data type that holds a sequence of characters that is treated as text.

`string` *ParameterName* : *Direction*

`string` *VariableName*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected when defining fields, global variables, and component variables.

Description

Uniface uses UTF-8 as its internal character set, so the `string` can contain Unicode characters. String data can include numbers and other numerical symbols but is treated as text.

A string value is enclosed in double quotation marks (`" "`).

```
variables
 string vDepartment, vManager
endvariables
```

Related concepts

- String Constants
- Packing Codes for Other Data
- Substitution in String Values
- Extracting Values From String Data

Syntax Strings for Pattern Matching
Case Conversion

struct

Uniface data type containing an ordered collection of references to one or more Structs.

{`byRef`} | {`byVal`} `struct` *ParameterName* : *Direction*

`struct` *VariableName* | *NonDatabase_FieldName*

Qualifiers

- `byRef`—the Struct is passed by reference, so only a memory pointer is passed, not the actual data, which is already available in memory.
- `byVal`—the data is passed by Value, so a copy of the Struct is created and then passed.

Use

Allowed for non-database fields, local variables, component variables, global variables, and general variables, and for parameters in ProcScript functions and partner operations.

The `byRef` and `byVal` qualifiers are only applicable for parameters. They specify how Structs are passed back and forth. They are not allowed for fields or variables of any kind.

Description

The `struct` data type is used for complex data that is typically represented as a tree-like structure consisting of nodes (sub-trees or branches) and leaves (data values). It is intended for data manipulation rather than storing data, so it is available for variables (local, component, global and general), parameters, and non-database fields.

Related concepts

Struct Variables
Passing Struct Parameters

Related reference

Structs

time

Uniface data type that holds a sequence of numbers that represent a time, in the format *hhnnss*.

`time` *ParameterName* : *Direction*

`time` *VariableSpec*

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can be selected

when defining fields, global variables, and component variables.

Description

The `$clock` ProcScript function can be used to convert data to the `time` data type.

```
params
 time pTimeStamp : IN
 lineartime pElapsedTime : OUT
endparams
pElapsedTime = $clock-pTimeStamp
```

Related concepts

- Date and Time Data
- Packing Codes for Date and Time Data

Related reference

- Data Handling in ProcScript

xmlstream

Uniface data type containing well-formed XML.

`xmlstream` [`DTD:` *DTDName*] *ParamName* `:` *Direction*

`xmlstream` *VariableName*

Parameters

DTDName—string, component constant, component variable, or global variable that evaluates to the name of the DTD that defines the structure of XML stream. *DTDName* has the format:

LiteralDTDName{`.` *LiteralModelName*}

LiteralDTDName is the literal DTD name defined in the application model specified by *LiteralModelName*.

Use

Use in `params` and `variables` blocks to define the data type of a parameter or variable. The data type can also be selected when defining fields, global variables, and component variables.

Description

XML data can be transferred between the `xmlstream` parameters and the component's external data structure with `xmlsave` and `xmlload`. In this case the XML must conform to the Uniface DTD for component data.

The `xmlstream` data type can also be used when transforming data from and to Structs using `xmlToStruct` and `structToXML`.

Public Operation with XML Stream

```
public operation getAccounts
params
  xmlStream [dtd:account.olb] x: inout
endparams
xmlLoad x, "dtd:account.olb"
retrieve/e "accountSsv.olb"
xmlSave x, "dtd:account.olb"
end; operation getAccounts
```

Related concepts

- Uniface XML Constructs
- Declarative XML Handling
- Structs for XML Data
- xmlsave
- xmlload
- xmlToStruct
- structToXml

Related reference

- XML Streams
- APIs for XML Handling

