Rocket software

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

operation

Declare an operation ProcScript module.

{`public` | `partner`} `operation` *OperationName* | *PredefinedOperationName*
  {`public soap`}
  {`public web` | `partner web` }
  {*ScopeBlock*} ; DSPs only
  {*ParamsBlock*}
  {*VariablesBlock*}

  Your ProcScript
{`end`}

Qualifiers

Table: Qualifiers

Qualifier	Description
`public`	Includes the operation in the component signature so that it can be invoked from an external component by an `activate` statement; default.
`partner`	Excludes the operation from the signature; it can only be invoked from within the component itself.
`public soap`	Operation can be called from a SOAP client when used in Service components, DSPs, and USPs. For more information, see soap.
`public web`	Operation can be called from a web browser, RESTful service, or other web client when used in DSPs, USPs, and Service components. For more information, see web.
`partner web`	DSPs only. Operation cannot be called from a web client, but data in the scope block can be included in the request-response exchange with other operations or triggers.

Parameters

Table: Parameters

Parameter	Data Type	Description
OperationName	Literal	Literal name of the operation; maximum length of 32 bytes. The characters can be letters (A-Z), digits (0-9), and underscores (_), and must begin with a letter (A-Z).
PredefinedOperationName	Literal	One of the following predefined operations, which define component behavior in specific circumstances: • exec—executed when the component is activated. It is the default component operation. • init—executed when a component instance is created with `newinstance` or `activate`. • cleanup—executed when a component is removed.

Parameter	Data Type	Description
		• attach—executed when a contained DSP is attached to a parent DSP. • detach—executed when an attached child DSP is detached from its parent.
ScopeBlock	Literal	DSP only. Specifies the data to be included in a DSP request-response exchange. For more information, see scope...endscope.
ParamsBlock	Literal	Defines the operation's parameters. For more information, see params...endparams.
VariablesBlock	Literal	Defines the local variables used by the operation. For more information, see variables...endvariables.

Return Values

The `operation` declaration itself does not return a value. However, you can use the `return` statement to have the operation return a value in `$status`.

Use

Allowed in all component types.

Description

The `operation` block defines an operation that can be called by an `activate` statement.

It acts as an implicit `end` for the previous operation or ProcScript module.

By default, an operation is `public`, meaning that it can be activated by other components. If you want it to be accessible only within the component, you must declare it as `partner`.

If you define scope, parameters, or variables for the operation, they should be declared in that order—first the `scope` block (in DSPs only), followed by the `params` block, and then the `variables`.

Operation Names

When specifying the operation name:

- Do not enclose the name in double quotation marks (").
- Do not use the names `accept` and `quit`. These are predefined operations that are expose the *accept* and *quit* triggers in the component signature.
- Do not define operations named `abort` or `complete`, because these are reserved.

Operations on the Web

When `operation` is used in web applications:

- Use the `public web` declaration if you want the operation to be activated from a web browser or RESTful web service.
- Use the `public soap` declaration if you want the operation to be activated by a SOAP-based web service.
- In DSPs, use the `partner web` declaration if the operation is to be referenced by scope definitions of triggers or other operations that are themselves declared as `public` or `partner web`.
- In DSPs, use a `scope` definition to define the data that will be included in the request-response exchange between the client and the server. If omitted, the scope is assumed to be both `input` and `output`, meaning that all data in the DSP will be included in both the request and the response.

Activating operations from the browser initiates a request-response cycle, since operations can only be executed on the server. To execute an operation on the browser, it must be defined with the `weboperation` command.

If an `operation` is defined with the same name as the `weboperation`, the last one defined is the one that is used.

Example: Defining an Operation

The following example shows the operation `DISCOUNT` of a service component named SERV1:

```
operation DISCOUNT
params
  string CUSTID : IN
  numeric AMOUNT : INOUT
  numeric PERCENTAGE : OUT
endparams
; no discount till proven otherwise
; 20% discount for Uniface
; 15% discount for Acme
; adjust amount
PERCENTAGE = 0
if ( CUSTID == "ufbv" ) PERCENTAGE = 20
if ( CUSTID == "acme" ) PERCENTAGE = 15
AMOUNT = AMOUNT * ( 100 - PERCENTAGE) / 100
end
```

The operation `DISCOUNT` could be referenced from another component as follows:

```
activate "SERV1".DISCOUNT (ID.CUST, TOTAL.INVOICE, $DISCOUNT$)
```

Example: Calling an Operation Recursively

The following example shows the operation FACTORIAL, defined in the Operations trigger of a service component named CALCULATOR:

```
operation FACTORIAL
params
 numeric N : IN
 numeric F : OUT
endparams
variables
 numeric W
endvariables
```

```
if ( N > 1 )
 W = N - 1
 activate "calculator".FACTORIAL (W, F)
 F = N * F
else
 if ( N = 1 )
 F = 1
 else
 F = 0
 endif
endif

end ; operation FACTORIAL
```

Related concepts

- Public and Partner Operations
- return
- weboperation...end
- operation init
- operation cleanup
- operation attach
- operation detach

Related reference

- Script Container