User's guide

Version 5.11

February 2025

Welcome to Renque

Renque is a powerful software environment designed for general-purpose discrete event simulations. It has a comprehensive graphical user interface and a versatile simulation engine. Renque allows you to build accurate simulations rapidly, for a wide range of applications.

The phrase simulation generally refers to a technique of imitating the behavior of some process or system by means of operating a model. Simulations are used to acquire numerical information and improve understanding of systems, in an environment where experimentation with the real system is undesirable or impossible.

Renque is a software tool developed to create and operate discrete event simulation models. A discrete event is something that happens in an instant of time, with zero duration. Although gradual system transitions can be represented in a Renque project, the program was designed primarily to deal with instantaneous changes.

Renque is a full-fledged *Microsoft Windows* application, suitable to analyze a virtually unlimited range of simulation objectives. Among the many supportive features that make modeling with Renque efficient and enjoyable are a template repository, advanced animation tools, a simulation clock with calendar, and a helpful event viewer. Renque also has a built-in script engine which makes it possible to tailor every detail of a simulation model precisely without limitations.

In this document, the names of user interface controls are displayed in bold, where suitable. The names on Renque object properties are written with a capital first letter, and property values are generally displayed in italic. Internal document hyperlinks are indicated by an alternative text color.

Licensing and support

Renque can be used free of cost for any purpose at no monetary cost for an unlimited period of time. Purchase of a Renque pro license removes a constraint in the maximum project size and grants access to Renque's excellent user support service for one year. The software is available for download on the Renque website www.renque.com.

System requirements

Operating system: *Microsoft Windows*™ 7 or higher 64-bit.

Contents

1. Getting started

This section is intended to familiarize new users with the Renque simulation environment. It contains an introduction to the user interface and two tutorials:

- Quick start guide — Introduces the Renque user interface and the principles of simulation modeling in Renque.
- Tutorial — Demonstrates the construction and operation of a simple project for a small supermarket.

1.1. Quick start guide

This topic contains an elementary tutorial that introduces the main features of Renque. You will visit and operate the most important elements of the user interface as you are guided through the construction of a very simple simulation model.

When Renque is started it briefly displays a splash window followed by the appearance of the primary user interface window, recognized by the project name in the title bar of the window. Models are constructed and simulation runs are controlled and viewed on the viewport, which is the large central area on this window. All other modeling and simulation controls are either located on the main window or accessible from it.

A Renque simulation has the three key elements: servers, links and entities.

- Server objects act mainly as container and processor of entities. They are represented by static pictures in the viewport. Servers are created on the viewport with the help of the mouse.
- Links function as means to transport entities between servers and to create and destruct entities. Links are also created by mouse operation in the viewport. They are displayed by curves.
- Entities are dynamic items in the project. As for servers, they are displayed as pictures on the viewport. However, entities are created and destructed during the simulation process, and they move around the viewport.

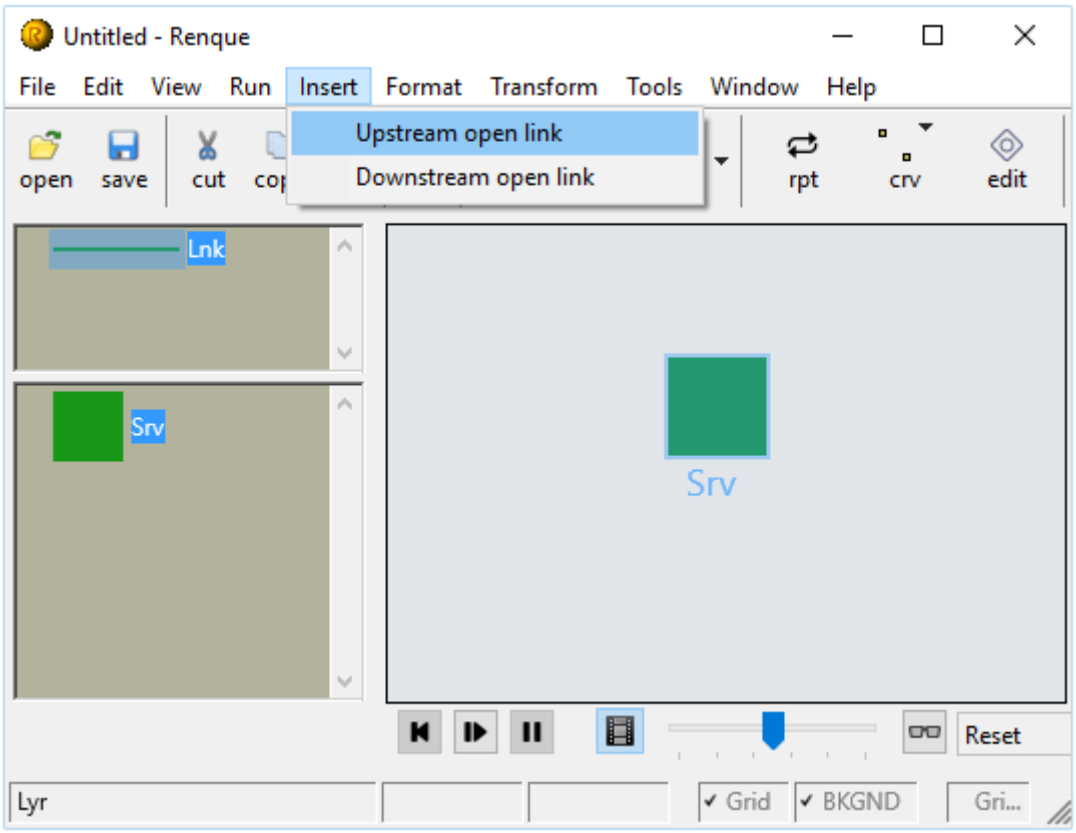
All project elements may or may not represent physical things found in the real world. If so, servers typically represent a process that takes some time to complete, such as a machine operation, storage of goods or physical transport, whereas links are better suited to symbolize transitions between states. Entities are more commonly seen representing mobile or temporary objects such as materials or documents.

The available server templates are found in the Toolbox on the Main window, located on the left side of the viewport. To add a server to the project, simply use the mouse to drag the template from the Toolbox and drop it on the viewport. Links are created directly in the viewport by connecting two servers with the mouse.

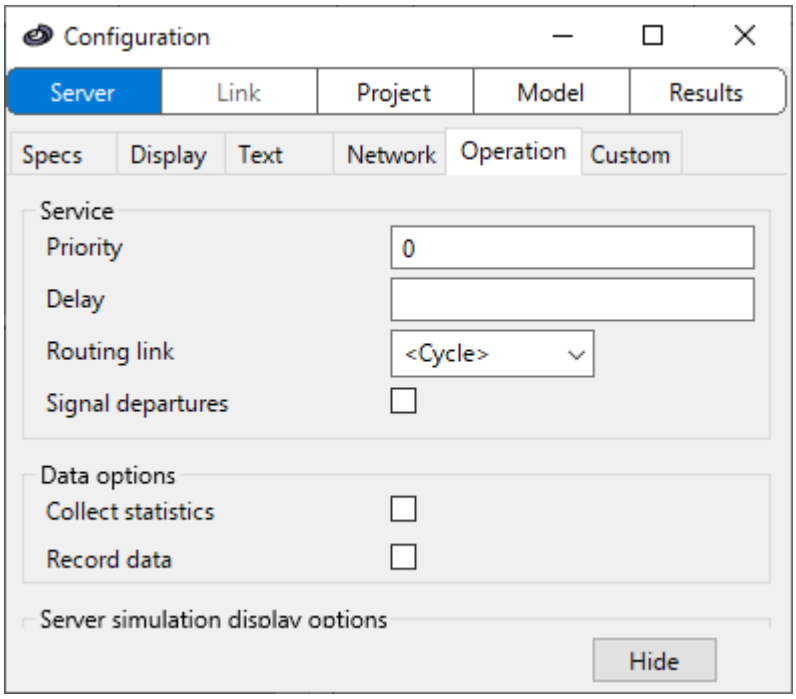
You can create a working project as shown in the figure below in the following three steps:

1. Create a server on the viewport by dragging the **Srv** template from the Toolbox and dropping it on the viewport.
2. **Select** the server with a mouse click.

3. Select the Insert menu command **Upstream open link** to connect a link to the server.



4. Double-click the server on the viewport. A new window titled *Configuration* is opened. This window is the hub to view and modify the properties of objects in the viewport.

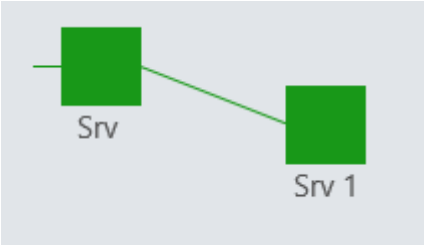


Select the **Operation tab** in the tab strip and enter the value *1* for the **Delay** property of the server. This property instructs the server to retain arriving entities for a single time unit before releasing them. Also check the checkbox labeled **Signal departures.** Application of this option causes creation of another entity when an entity leaves the server. The other controls on the tab represent server properties that are not of primary concern for the present demonstration. The section Object configuration provides further details.

That's it: A Renque simulation model. Now, you can run a simulation by clicking the **Restart** button in the simulation control toolbar underneath the viewport. The moving circle-shaped icons represent entities, travelling to the server after having been created on the link. Notice the simulation time advancing in the right-most pane of the Simulation control toolbar. The animation speed can be altered by the **Animation speed slider** on the same toolbar.

The link creates an initial entity at the simulation start. When this entity arrives at the server on the other link end, the server stores the entity and retains it for a period of *1* time-unit. After this period the entity is destructed because the server has no downstream links. When the entity has been released by the server a new entity is created on the link and the process is repeated. As a result, the server produces a stream of entities, with the interval time determined by the Delay property. A detailed description of the simulation mechanism is given in the reference section.

Interrupt the simulation by clicking the **Pause** button adjacent to the restart button. Next you are invited to extend the project with another server from the template in the toolbox. Add another server to the **viewport** somewhere to the right of the previously created server. Create a link between the two servers as follows: Click on a link template in the **Link toolbox** to activate link creation. Move the mouse over the right edge of the ***Srv*** server until a triangular connection location is shown. After clicking on the connection point, a line appears that will follow the motion of the mouse. Move the mouse over the left edge of the ***Srv1*** server and verify that another connection point appears. Now, click on the second connection point to complete the new link, which is represented by a line connecting the two servers. After creation of the link, the viewport should look somewhat like this:



Restart the simulation. In the animation you can see that the entities travel between the two servers. On arrival at Srv1 the entities are stored in the server object. They are shown on a horizontal curve on the server image. The number of stored entities that are displayed is limited by how many of the resident entity icons fit on this curve. The counter above the server image indicates the actual number of entities stored. The entities are not released from the server because the Delay property of Srv1 has the default value *Nil*, which makes the server behave as a classic queue of unlimited capacity.

Renque also has a number of object types that are not displayed on the viewport. These objects are referred to as components, and can be applied to expand the

capabilities of a project in various ways: Attribute and variable components are used to store data, distribution components provide random variation to a project, and schedule components perform actions at specified time points in a simulation.

A simulation project relies heavily on the application of Customization scripts. Customization offers the designer an all-purpose tool to model any imaginable aspect of a logical system by adaptation of the simulation engine using scripts. For example, in order to understand how the server named *Srv* creates an entity stream, double-click this server and select the **Custom tab** adjacent to the Operation tab on the Configuration window. The text edit area on the right displays the script:

```
if Not .ArrivalAccept then .EntityProcure
```

This is the default customization script for servers. The server script is executed each time an entity arrives at the server. If the aforementioned Signal departures option is applied the server script is also executed when an entity leaves the server.

The dot characters preceding the script method names in the script syntax indicate an implicit reference to the server for which the script is being executed. The `ArrivalAccept` function call invokes processing of an entity arriving at Srv. Without this function call the entity arrival would be ignored and the entity remains in the link. The `ArrivalAccept` function is ineffective when executed for a departing entity, in which case it returns the value *False.* As a result, the `EntityProcure` is called, which creates a new entity on the upstream open link of Srv. As this is done for every entity departing from the server, a cycle is created which produces a stream of entities.

The first entity arrival is seeded by the default customization script of the upstream link, which is executed once at the simulation start. The link customization may be inspected by double-clicking the link in the viewport and selecting the Custom tab on the Link console.

This next section in the document contains a tutorial, which is intended to familiarize the user with the most important user interface items. The tutorial demonstrates the application of customization scripts and components and also introduces some commonly applied simulation modeling practices.

1.2. Tutorial

This tutorial comprises construction and operation of a simple project for a small supermarket. The level of detail in the description of the required user actions presumes that the reader is familiar with the content of the Quick start guide section of this document. The tutorial demonstrates the application of some server and link properties and the interpretation of simulation results within a practical context. Also, it introduces the application of components to add random variation to the project and to store data in entities. In addition, it will explain how to use customization scripts in a project.

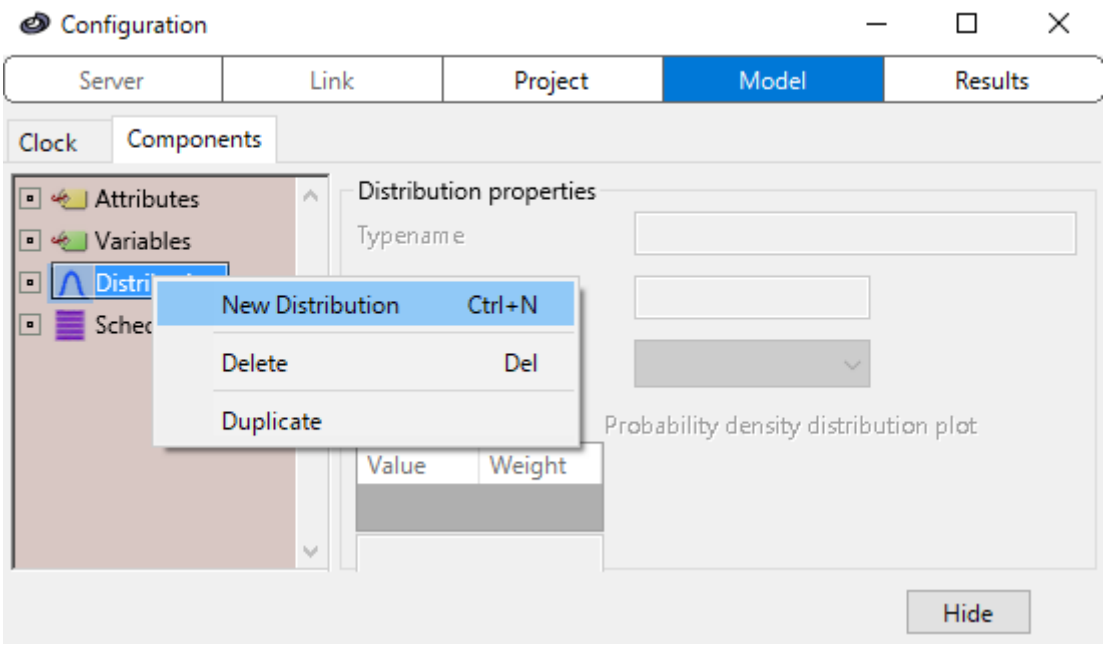
Imagine a supermarket where costumers arrive, spend some time shopping, choose a cash register, wait their turn, and then check out and leave. Suppose the manager of the supermarket would like to know more about how the number of cash registers affects the waiting time of the costumers for the check-out routine. This tutorial demonstrates how to go about in building a Renque project for the supermarket and extract this kind of information from it.

Start the Renque application and create the project displayed in the figure, consisting of four server object and five links. The server names are assigned by double-clicking a server and typing the desired text into the Name property text field of the Server console of the configuration window.

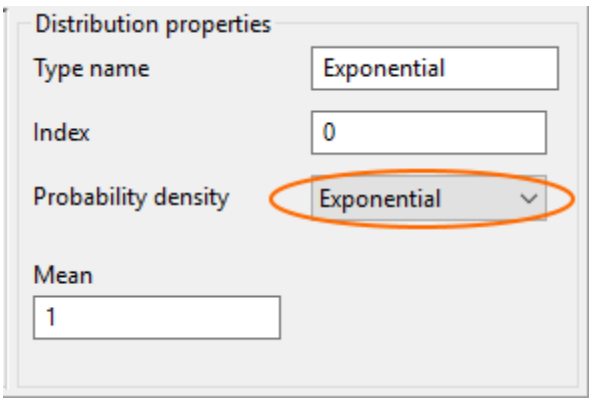
The entities in the stream leaving the Arrival server represent the customers arriving at the supermarket in the model. The customer entities are sent to the Shopping server where they are stored for a specified time period. The storage time represents the time spent shopping by the customers. The Shopping server subsequently sends the entities to the Queue server, where they will be stored until removed by the Checkout server. The Checkout server is used to implement the check-out process by application of another delay. Finally, the entities are sent to the open downstream link, which represents the departure of the customers from the establishment by destructing the entity.

The Arrival server will release a stream of customer entities with an interval time equal to the Delay property of the server. The customer interarrival times at the supermarket will show random variation. The inter-arrival times of such processes as customer arrival in a shop are best described by an exponential probability density function. Renque is capable of generating random variation by means of distribution components.

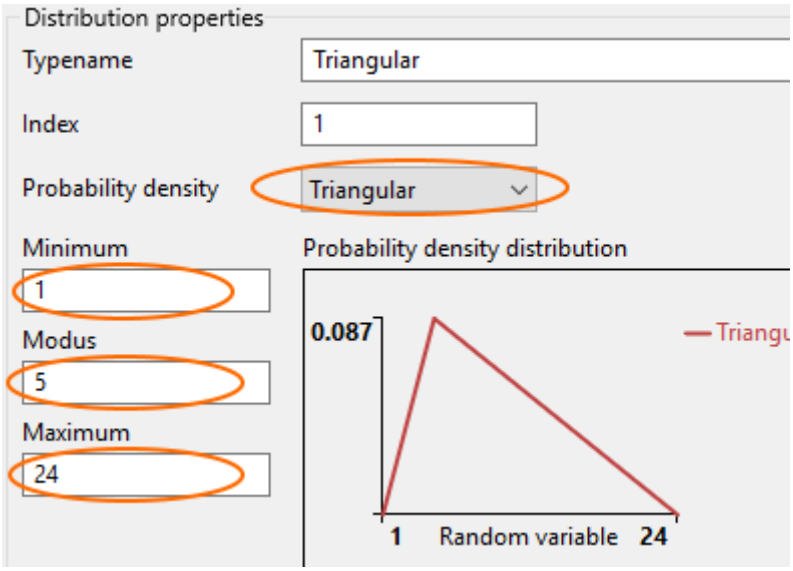
Creation of distribution components
A distribution component that has the required exponentially distributed variation is created as follows: Open the Configuration window by the corresponding command in the Tools menu and select **Model console** at the top of the window and subsequently the **Components tab** on the second row. Select and right-click the **Distribution** item in the repository on the left and select the **New Distribution** command in the contextual menu.



This command creates a new distribution component, which appears as selected item in the Component repository. Change the **Probability density** function of the component by selecting the *Exponential* item in the drop-down menu. Keep the default value *1* for the Mean property.



Following the same procedure, create **another distribution component** of type *Triangular* with properties (min, mod, max) = (1, 5, 24), which will be used for the shopping time. The parameter values must be assigned in the reversed order: max, mod, min to prevent the interface from rejecting an invalid entry. The chart at the right side of the window displays a plot of the probability density distribution.

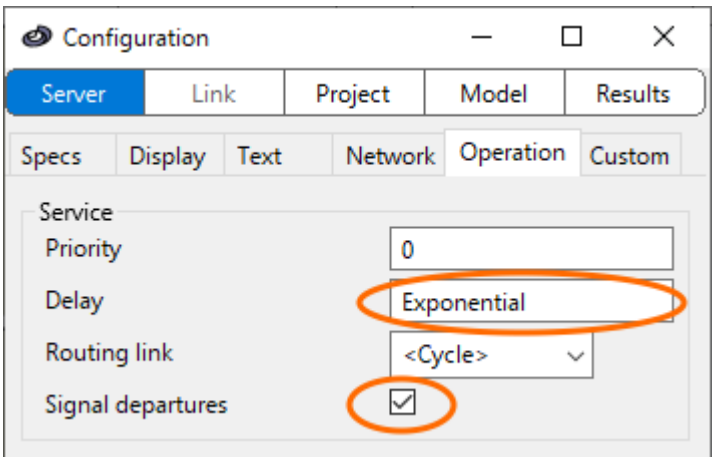


These two distribution components can be assigned to timing-related object properties, such as the server Delay property. The component produces random numbers distributed in accordance with the component properties, using a built-in random number generator.

The next step of this tutorial involves adjustment of server properties to obtain the desired behavior of the model.

Preparation of the Arrival server

Return to the Workspace window, double-click the ***Arrival*** server and assign the *Exponential* distribution component to the **Delay** property on the **Operation tab** by typing the component name in the text field or pressing **Shift+space** key and selecting *Exponential* in the pop-up list. Also check the **Signal departures** box and press the **OK button**.



You have assigned the distribution component named Exponential, of type *Exponential,* with parameter *Mean* = *1*, to the Delay property of the Arrival server. As a result, each entity stored in the server is assigned a randomly generated delay time value by the distribution component.

Preparation of the Shopping server

Return to the Workspace window, double-click the ***Shopping*** server and Assign the *Triangular* distribution component to the **Delay** property of the Shopping server. The Shopping server can process an unlimited number of customers simultaneously, which will be all be retained in the server for a period dictated by the *Triangular* distribution. Also *check* the **Signal departures** box and select the value *<Random>* for the **Routing link** property. Press the **OK button**

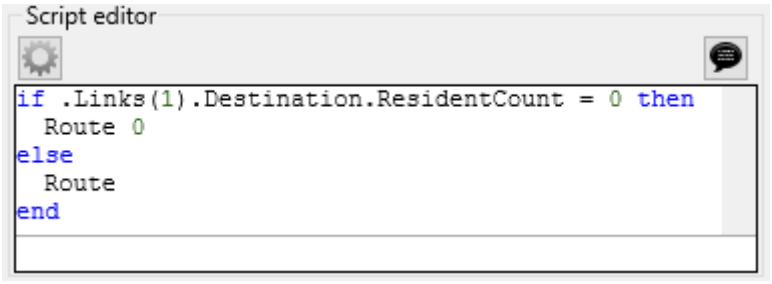
Preparation of the Queue server

Return to the Workspace window and double-click the ***Queue*** server. The Queue server needs to pass an arriving entity directly to the Checkout server if this server is empty. This behavior is accomplished by a short script. Click the **Custom tab** on the Server console to access the script editor. The editor is a versatile tool for composing scripts, with a variety of useful features such as syntax checking and code autocompletion.

Type or paste the following script lines into the text area of the editor, replacing the default script line shown in the editor:

```
if .Links(1).Destination.ResidentCount = 0 then
      Route 0
else
      Route
end
```

When completed, the editor should look like this:



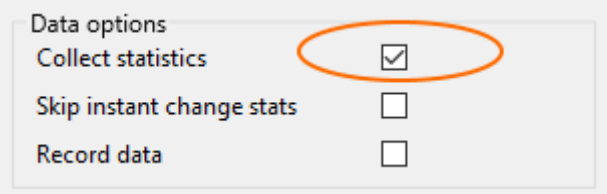
The first line checks if the Checkout server is empty, using the `ResidentCount` function. If so, the `Route` method on the second code line stores the arriving entity in the server and routes it with delay time *0* to the Checkout server. If the Checkout server is not empty, the Route method without arguments stores the arriving entity in the server for an indefinite period of time without routing to a link, because the default value for the Delay argument is *Nil*. The Route call operates implicitly on the arriving entity because no entity identifier is given. Note that the script is executed for entity arrivals only and not for departures, because the Signal departures option is not applied to the Queue server.

Preparation of the Checkout server

The Checkout server represents the check-out routine. **Double-click** the Checkout server. First, check the ***Signal departures*** checkbox, which activates the emission of

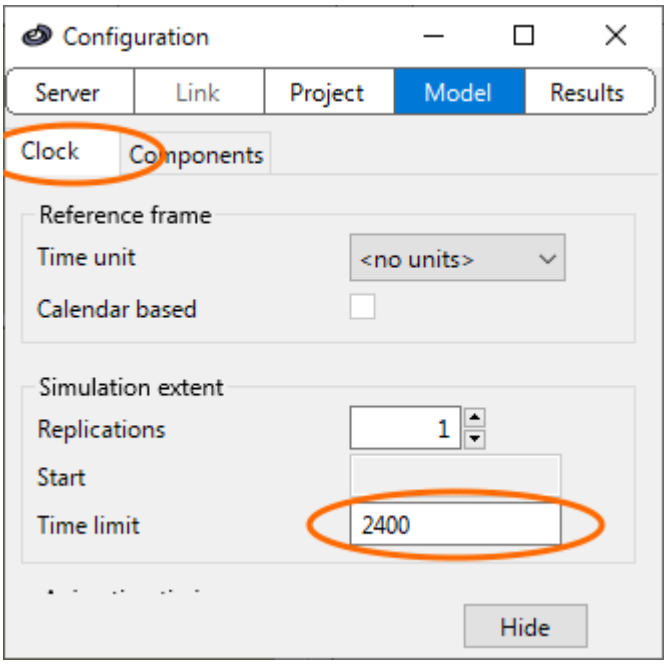
a departure signal when an entity leaves the server. Then type a value *0.9* in the **Delay** property text field. In reality the checkout time will also show random variation and there will be a correlation with the shopping time. However, a uniform checkout time is assumed for now. The value *0.9* renders the customer processing capacity at the check-out 10% higher than the average customer arrival rate, imposed by the mean value *1* of the Exponential distribution of the Arrival server Delay property. As a result, the Checkout server is expected to be occupied 90% of the time.

Finally, select all objects in the viewport, open the Configuration window, select the **Operation tab**, and check the **Collect statistics** checkbox in order to activate statistics collection for the servers.



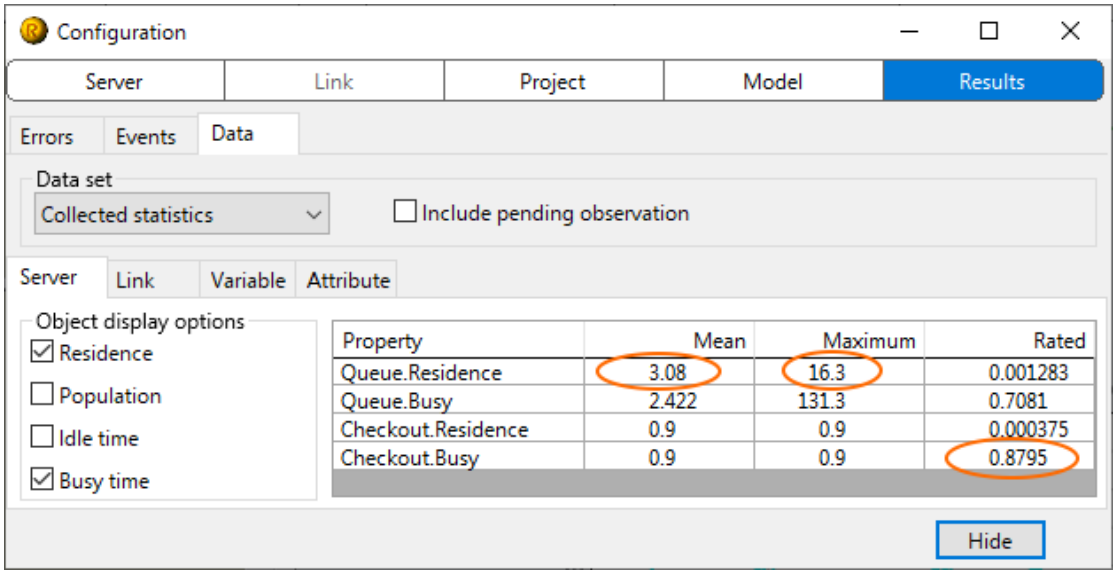
Inspection of simulation results

Before running the project, a maximum run time is set for the simulation. We assume that the time unit for the project is minutes. Select the **Clock tab** of the **Model category** of the properties window and enter the value *2400* (5 days in office hours) in the Time limit text field. For the present project it in not necessary to select the *Minutes* value for the Time unit property.



The simulation project for the supermarket, although still quite rudimental, is now ready to be run. Press the **Restart** button and watch the Shopping server initially filling up to some 10 customers, after which the first customers start arriving in the empty Queue to check out and leave. After leaving the project running for a while, the number of costumers shopping and queuing start to vary continuously.

Turn the **animation mode off** and let the simulation continue until it stops. **Select** the Queue and Checkout servers in the viewport and select the Results console at the top of the configuration window. On the Data tab select *Collected statistics* from the **Data set** pull-down menu. A data sheet appears containing the collected statistics for the selected servers.

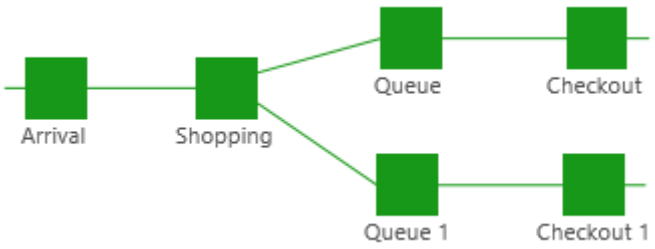


In the *Queue.Residence* row of the data sheet, the column labeled **Maximum** reports a maximum residence time of 16 minutes. The reported **Mean** is 3.1 minutes. Also note that Checkout server is occupied for about 88% of the time, as indicated by the **Rated** item in the *Checkout.Busy* row, which represents the fraction busy status time of the total simulation time. This agrees with the expected value of approximately 90% discussed above.

Obviously, the queuing times observed will be unacceptable to the clients of the supermarket. The high maximum results from the high standard deviation of the exponential distribution applied to the Delay property of the *Arrival* server, which causes busy next to quiet periods in the supermarket. As the check-out processing time is fixed, customers queue up during the busy periods.

Model expansion

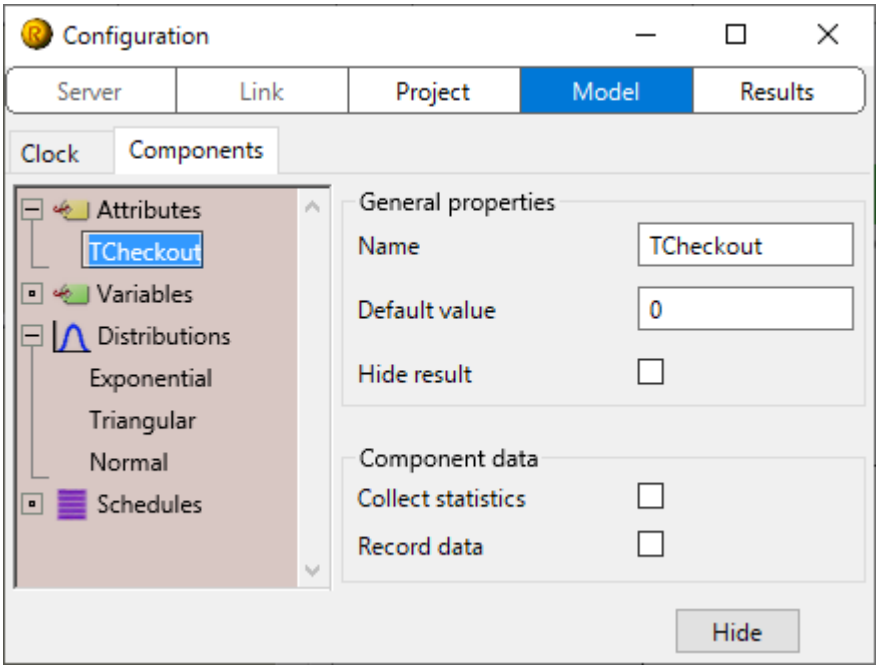
The next step of the tutorial demonstrates how to increase the number of cash registers in order to reduce the customer queuing times. The simplest way to increasing the check-out server processing capacity is to increase the Capacity property value. However, this method is unsuitable because each Checkout server has its own queue. Instead, select all server and link objects to the right of the shopping server. Then, while keeping the **Ctrl key** on the keyboard pressed, **drag** the Checkout server downwards with the left mouse button to duplicate the selected objects. After a simulation reset the result should look similar to this:



Before the model is run again to verify the effect of the additional check out line, some refinements are made to render the model more realistic.

Checkout time refinement

The next exercise in this tutorial involves implementation of a checkout time that depends on the time spent shopping by a customer, with random variation. Create a new attribute component by selecting the **New/Attribute** item in the pop-up menu **Component repository**. Rename the new attribute *TCheckout*.



An attribute component allows storage of data in individual entities. The attribute will be used to store a pre-determined checkout time for a customer into the entity that represents the customer. Select both **Checkout servers** in the viewport by dragging a rectangle from top-left to down-right enclosing only the two servers. Double-click one of them to open the Configuration window once again. Select the **Operation tab** at the top of the window and assign the new attribute component TCheckout to the Delay property of the selected servers by typing *TCheckout* into the **Delay** text field or using autocompletion with the **Shift-space** key.

Also, create a **new distribution** component of type *Normal* and keep the default parameter values.

Double-click the ***Shopping*** server. First, select the **Operation tab** on the Server console and check the **Signal departures** option. Next, select the **Custom tab** and place the following script in the script editor:

```
If not .ArrivalAccept Then
 Dim ShoppingTime as Double = Sim.Time - EntryTime
 Normal.Parameter(1) = 0.12 *  ShoppingTime
 Normal.Parameter(2) = 0.024 * ShoppingTime
 TCheckout = Normal.Evaluate

 if Queue.ResidentCount < Queue1.ResidentCount then
     Transfer .Links(1)
 elseif Queue.ResidentCount > Queue1.ResidentCount then
     Transfer .Links(2)
 end if
End if
```

The first line performs routing on an arriving entity using properties assigned to the server. The top four indented script lines assign the two parameters of the Normal distribution component and store a generated random number into the Tcheckout attribute of the departing entities. The `EntryTime` function call returns the arrival time of the departing entity in the server. The difference with the simulation time corresponds to the shopping time. Both the **Mean** (Parameter index = 1) and the **Standard Deviation** (Parameter index = 2) of the normal distribution are assigned a value proportional to the shopping time. As the TCheckout attribute has been assigned to the Checkout Delay property, the checkout time for each customer will be a normally distributed random value with a mean and spread proportional to the shopping time spent by that particular costumer. Unlike Renque variable components, the locally declared ShoppingTime script variable exists only within the present customization script.

The mean shopping time, imposed by the Triangular distribution is (1 + 5 + 24) / 3 = 10 minutes, such that the overall mean checkout time for all costumers is 10 * 0.12 = 1.2 minute, which leads to an expected occupation of the two checkout servers of 1.2 / 2 * 100% = 60%.

Queue selection refinement

This script also improves the mechanism for queue selection by the customers in the model. In the real supermarket we expect customers to join the shortest queue when they arrive at the check out area. The last part of the script makes sure that the departing entity is sent to the queue server with the least number of residents. The `Transfer` command removes the departing entity from the server and sends it directly to the link passed as argument to the command. If both queues have an equal number of residents the destination server is randomly selected as a result of the value <*Random*> assigned to the Routing link-property of the server.

Simulation results

Run the simulation again and have a look at the results. The reported average queuing time is under a minute, with an observed 6 minute maximum over one week of operation.

Property	Mean	Maximum
Queue.Residence	0.7811	6.654
Queue1.Residence	0.7602	5.921

The behavior of the queues in time can be inspected in the chart below, which was created by the Renque charting feature. The curves clearly show for a representative time segment of the simulation that the queue population patterns do not differ much between the two queues, For the sake of brevity, a detailed guide to Renque charting

is saved for a later occasion. The tutorial file with chart included can be downloaded for examination from our website.

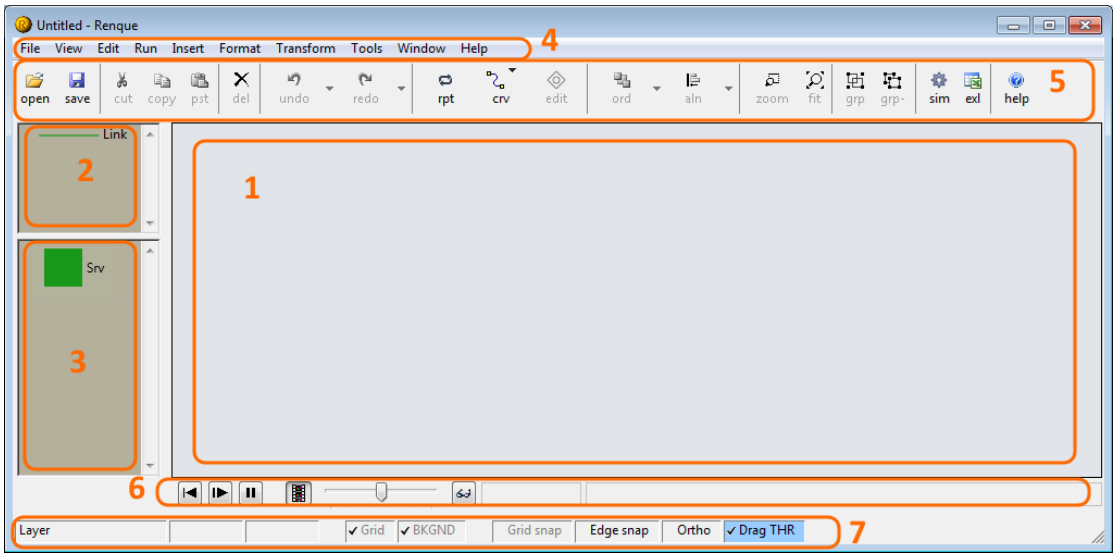
Conclusion

Simple as it may seem, the tutorial project for the supermarket is quite realistic, although the data applied to the project parameters are fictitious. In order to reach a usable level of accuracy, these parameters need to be derived from data acquired from observations in the real supermarket or from suitable estimation methods.

One conclusion that the manager might come to by using the simulation is that substantial cash register overcapacity is required to prevent losing clients to competitors on frustration about long waiting times. One obvious strategy to increase employee utilization is to assign cash register operators to other duties during quiet periods, such as shelf stocking.

2. The application workspace

Renque simulation models are constructed and operated in the workspace window. The workspace window is the principal user interface of the application. It is opened when the application is started, and the program exits when the workspace window is closed.



This section starts with a discussion of the Application properties of the application and has separate topics for each of the elements of the workspace window:

1. Viewport — The model construction environment.
2. Link toolbox — Contains templates for link creation.
3. Server toolbox — Contains templates for server creation.
4. Menu bar — Categorized list of commands.
5. Design toolbar — Buttons for frequently used menu bar commands.
6. Simulation toolbar — Controls simulation operation and displays information on the state of a simulation.
7. Status bar — Displays the current layer of the viewport and controls various editing options.

Additional characteristics of the behavior of the workspace are given in topics:

- Layers
- Blocks and fusions
- Curve types
- Viewport pages

2.1. Application properties

Renque is a stand-alone application. Multiple instances of Renque may be started in the operating system, but you can only operate a single project in a Renque instance.

Projects are created, opened and saved by the usual commands **New project**, **Open project**, **Save project** and **Save project As**, which are available in the File menu of the Main window. Double-clicking a file with a Renque-associated file extension in Windows Explorer will open the file in a new Renque instance. Drag-dropping a file from Windows Explorer onto the viewport area of the Main window will open the file, after prompting to save the project being replaced. A Renque instance is closed by the Exit command on the File menu or by clicking on the 'X'-button on the title bar of the Main window.

Renque project file types

There are three Renque project file types:

- The Renque project file (extension: *.rnqp)
- The Renque server toolbox file (extension: *. RenqueServerToolbox)
- The Renque link toolbox file (extension: *. RenqueLinkToolbox)

The Renque project file supports storage of all information that can be contained in a Renque simulation project, including picture components and all simulation data.

2.2. Viewport

A Renque simulation model is constructed and operated in the viewport area. It supports the following operations, using the mouse, the Menu bar, the toolbar, or the contextual menu of the viewport:

- Add and delete servers and links.
- Cut, copy & paste servers and links.
- Group/Ungroup objects.
- Merge objects into Blocks and Fusions.
- Rearrange servers.
- Resize and edit servers.
- Edit link curves and change link connections
- Change the display order of servers.
- Assign an object layer.
- Create link templates and server templates
- Operate simulations
- Create a page background

Server creation

Servers are added to the model by drag-dropping a template from the Server toolbox onto the **viewport**, using the mouse. Server objects added to the viewport are placed in the Active layer. They are represented by an image and a name tag.

Link creation

Link objects are depicted on the viewport by a curve and an optional name tag. Link creation is started by selecting a link template from the Link Toolbox. The finger mouse pointer indicates that the viewport is ready to receive instructions for construction of a new link object. If the mouse hovers over a connectable region of a server, the possibility for making a link connection is indicated by a change of the mouse pointer. The connection location on the server is indicated by a diamond or triangle marker. The connection is established by a left mouse button click on the connection location. The location at which a link connects to a server is called a knot.

After the link start point has been created, each click on the viewport extends the curve path by addition of a control point, indicated by a small circle. Link construction is completed by creating a termination point using the same routine as for the start point. New link objects created on the viewport are placed in the Active layer. The curve type of a new link object is determined by the selected **Curve type** item in the Format menu. The orientation of the last segment of a link curve of type orthogonal can be switched between horizontal and vertical by holding down the space bar while the mouse hovers over a (non-auto arranged) destination connection point.

Open link end points, which are not connected to a server, can be created by clicking anywhere on the viewport while holding the **Ctrl**-key pressed. An open curve end point is indicated by a large circle.

A server connected to the upstream side of a link is named the **origin server** of the link, while the link is referred to as a **downstream** link of the server. The server connected to the downstream side of the link is the **destination server** of the link, while the link is referred to as an **upstream** link of the server. Servers can have an unlimited number of connected links. Links that are not connected to a server on either or both sides are called **open** links.

Object selection

Selected servers and links are indicated in the viewport by a semi-transparent mask over the object in the selection color of the operating system. Selection of a single object is simply done by clicking on it in the **viewport**. The selection result depends on whether or not the **Ctrl key** and **Shift key** are pressed during the selection operation.

Shift key state	*Ctrl key state*	*Selection result*
released	pressed	remove from selection
pressed	released	add to the selection
released	released	start new selection

To select/unselect several objects at the same time: Click with the left mouse button on an empty location on the viewport and move the mouse while keeping the mouse button pressed. A rectangle is displayed to indicate the operative region. The selection operation is executed when the mouse button is released. The result of this rectangular selection depends on the horizontal sweep direction. If the mouse was moved from left to right, only objects that lie completely inside the rectangle are affected. If the mouse is moved from right to left objects touching the rectangle are selected as well. The effect of pressing the Ctrl and **Shift** keys for to this selection method is the same as for selection of single objects by mouse clicks.

Object rearrangement and duplication

The position of servers on the viewport can be changed by dragging their image across the viewport. If the control-button on the keyboard is pressed while starting the drag operation, the selected servers and selected links connected to them are duplicated and subjected to the drag operation. As opposed to the copy/paste menu commands, duplication by mouse drag operation does not store the objects in the clipboard of the operating system.

Object editing

The **Edit objects** command of the Edit menu toggles the edit mode for selected objects in the viewport. In the edit mode, bounding box sizing handles and curve control points are displayed which can be used to change the objects in the viewport with mouse operations. The following object elements are available for modification:

- Height and width of the server icon
- Contour of the server residents-curve
- Contour and connectivity of the link curve
- Tag anchor location of the link
- Size and location of the symbol text area
- Contour of the symbol curve.

Elements cannot be edited if they are hidden in the viewport for the present simulation state and selected Simulation preview option. Furthermore, the symbol curve of a server cannot be edited if the server resident curve is visible as well. Specific element types may be locked for all edited objects using the Edit options command of the Edit menu.
Resizing handles are displayed as small rectangles on the edges of the bounding box of the element. Curve control points are displayed as small circles and the link tag location is indicated by an encircled cross sign. These markers can be selected in the same manner as objects are selected on the viewport by mouse operations. Objects are modified by dragging the selected markers using the mouse. The aspect ratio of resizing actions is preserved, unless **Ctrl key** is pressed. Curve control points can be duplicated by dragging a selected control point while keeping the **Ctrl key**

pressed. Pressing the **Del key** removes the selected control points of the curve. The curve type of all edited server and link curves can be changed by selecting the desired **Curve type** item in the Format menu.

Links can be detached from connected servers by dragging the link end point away from the server. Reversely, links can be attached to a server by dropping the end point on a suitable connection knot of a server. The knot is indicated by a marker in the shape of a diamond or a triangle. A vertical line inside a circle distinguishes the termination point of a curve from the start point. A polygon or spline curve with more than two control points is fully rescaled if one of the end points is dragged in the viewport. Alternatively, if the space bar is held down at the start of the drag operation, only the curve end point is displaced. For orthogonal link curves, the orientation of the first and last segment of a link can be switched between horizontal and vertical by holding down the space bar while the mouse hovers over a link end point other than an auto arranged node connection knot. Pressing the **Esc key** terminates the edit mode without storing any changes made to the selected objects.

Viewport manipulation

The viewport may be scaled and translated by dragging the mouse. The viewport is translated if the mouse is moved while keeping the **right button** pressed. The viewport is dynamically scaled if the **Ctrl key** is held down at the same time. The same behavior is obtained for the left button if the **Pan button** on the Toolbar is toggled on. In that case the right button scales the viewport without pressing the Ctrl key and the mouse cannot be used for object selection.

The viewport may also be scaled by the **Extend viewport** command on the View menu and by the **Zoom button** on the Toolbar. The Zoom button enables scaling to a region specified by dragging the mouse. If the mouse is **dragged** in the Viewport after pressing the zoom button, a rectangle is drawn of the same aspect ratio as the Viewport, centered around the mouse-down location. The size of the rectangle corresponds to the viewport scaling factor applied when the mouse button is released. The scaling direction depends on the direction of the mouse displacement. If the mouse was moved from to the right the viewport scale range is decreased to the rectangle area scale range. If the mouse button was moved to the left, the viewport is expanded to the range that results from shrinking the Viewport area into the rectangle area. For a small mouse displacement, the viewport scaling factor remains unchanged and the center of the viewport is translated to the mouse location.

Viewport contextual menu

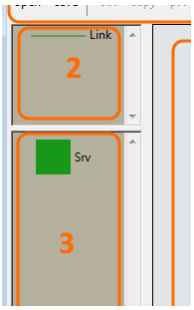
The contextual menu of the viewport is opened by a right mouse button click. The availability commands of this pop-up menu depends on the selected objects in the viewport and on the applied model properties. All commands that may appear in the contextual menu of the viewport are listed below, with a brief description of their function.

- **Cut/Copy/Paste** — Edit menu command clones
- **Editor options** — Clone of the corresponding options arrangement in the Edit menu.
- **Reset symbol text size** — Restores the default symbol text size of the edited servers of which all text area sizing handles have been selected in object editing.
- **Align to** — Performs the action of the corresponding Edit menu item with the selected server indicated by the contextual mouse click location as reference location.

• **Reset chart scale range**	Restores the default scale range of the selected chart server indicated by the contextual mouse click location.
• **Open fusion window**	Performs the action of the corresponding Format menu Blocks command for the selected block or fusion server indicated by the contextual mouse click location.
• **Create server template**	Performs the action of the corresponding Transform menu item for the selected server indicated by the contextual mouse click location.
• **Create link template**	Performs the action of the corresponding Transform menu item for the selected link indicated by the contextual mouse click location**..**
• **Move arranged link up**	Changes the auto-arranged connection order at the server for the selected link and link connection side indicated by the contextual mouse click location.
• **Move arranged link down**	Changes the auto-arranged connection order at the server for the selected link and link connection side indicated by the contextual mouse click location.
• **Insert page reference**	Performs the action of the corresponding Transform menu item for the selected link indicated by the contextual mouse click location, regardless of the page location of the link origin and destination servers.
• **Remove page reference**	Performs the action of the corresponding Transform menu item for the selected page reference server indicated by the contextual mouse click location.
• **Remove empty page**	Performs the action of the corresponding Transform menu item for an empty page indicated by the contextual mouse click location.
•	

2.3. Toolboxes

Toolboxes are used to create objects in the viewport. There is a toolbox for link templates (Item 2 in the figure) and a toolbox for server templates (Item 3 in the figure). The toolboxes may be hidden by unchecking the **Toolbox** item on the View menu.



The toolboxes contain one or several default templates. Server and Link objects in the viewport can be stored as new template in the toolbox by the **Create template** command from the Transform menu or the contextual menu of a selected object. A new template inherits all properties from the source object. Template properties cannot be changed.

Link toolbox

The link toolbox contains link templates. Links created on the viewport inherit all properties from the selected template in the link toolbox.

Server toolbox

The Server toolbox (Item 3) contains server templates. A new server object is created from a template by dragging a template from the toolbox and dropping it on the viewport.

2.3.1. Toolbox properties

The template images in the toolbox can be rearranged by drag-drop mouse operation.
The toolbox properties can be modified by commands of the contextual menu of the toolbox. Besides the regular **Cut**, **Copy**, **Paste** and **Delete** items, the Toolboxes have the following contextual menu item:

- Add category — Creates a new category for template categorization.
- Incrm. col. count — Adds a column for template organization in columns.
- Decrm. col. count — Removes the right-most column.
- Select children — Selects all viewport objects created by the selected template.
- Properties — Opens a window which allows modification of the toolbox properties.
- Config/Write — Saves the toolbox properties and content to disk.
- Config/Read — Reads the toolbox properties and content from disk.
- Config/Import — Adds toolbox content from disk to the present configuration.
- Config/Restore — Restores the default set of Toolbox templates.

2.4. Menu bar & Toolbar

The Menu bar is located right underneath the title bar of the Main window. It has a series of commands grouped into drop-down menus. The most frequently used Menu bar commands are also available as a button on the Toolbar, which is located under the menu bar.

The Menu bar commands are listed below, together with the corresponding Toolbar button icon, if existing, and a brief description of their function.

File menu

	New project	Starts a new project and discards the current.
	Open project	Opens a project file.
	Save project	Saves the current project.
	Save project as	Saves the current project after prompting for a file name.
	Page setup	Opens the Page setup selection menu.
	Print	Prints the project to a printer. Requires a completed procedure for the above-mentioned page setup menu item.
	Exit	Closes the application without saving.

Edit menu

	Undo	Undo the last mutation to the project. Clicking on the drop-down arrow opens a mutations list to select an item from.
	Redo	Redo the last undone mutation to the project. Clicking on the drop-down arrow opens a list of undone mutations to select an item from.
	Repeat	Toggles the Repeat mode for object creation
	Cut	Removes selected objects from the viewport and stores them in the Windows clipboard.
	Copy	Copy the selected objects the viewport to the Windows clipboard. Also available as pop-up command on the viewport.
	Paste	Paste objects from the Windows clipboard to the viewport. Also available as pop-up command on the viewport.
	Delete	Delete objects selected in the viewport.
	Edit objects	Toggles the viewport object editing mode
	Editor options	Toggles editing of specific server elements, including symbol curves and text area size, server size and link tag anchor location
	Find	Opens the Search utility to select servers by string comparison.
	Select	Selects specific object types in the viewport.
	Unselect	Unselects specific object types in the viewport.

View menu

	Simulation preview	Toggles the Simulation preview display mode
	Extend viewport scale	Rescales the viewport to fit the entire model.
	Scrollbars	Determines when the viewport scrollbars are displayed. Selecting the Auto option displays the scrollbars only if the model does not fit into the viewport.
	Toolbox	Hides the Toolbox when unchecked.
	Toolbars	Hides a specific toolbar when unchecked.

Run menu

	Reset	Resets the simulation run, discarding all simulation data.
	Restart	Starts a new simulation run, discarding all simulation data of the current run.
	Pause	Pauses or resumes the current simulation run.
	Animation	Toggles animation on and off.
	End replication	Ends the current replication, preserving all replication data.

	Reset replication	Resets and discards all data of the current replication.

Insert Menu

	Upstream open link	Adds a upstream link to all selected servers on the viewport. The server is the destination server of the new link. The link is open on the upstream side.
	Downstream open link	Adds a downstream link to all selected servers on the viewport. The server is the origin server of the new link. The link is open on the downstream side.

Format Menu

	Align	Allows alignment of selected server objects.
	Order	Allows adjustment of the server z-order. Newly created servers are placed on top of existing servers.
	Group objects	Joins the selected objects in the viewport into a group, forcing subsequent joint selection of all group members.
	Ungroup objects	Breaks up all selected groups in the viewport.
	Curve type	Controls the curve type for new links and for curve editing. The selected type is reflected by the menu icon. The icon with no line indicates the link template default or the value <*various*>.
	Blocks	Manages blocks and fusions.

Transform menu

	Invert links direction	Inverts the direction of the links selected on the worksheet
	Make link chain unidirectional	Realigns the direction of all of all selected links, provided the links selected form a continuous trajectory.
	Swap link name	Swaps the name properties of exactly two selected links
	Reset symbol text size	Adjusts the text area of symbol servers to fill the symbol shape.
	Create link templates	Creates templates in the toolbox for all link objects selected in the viewport.
	Create server templates	Creates templates in the toolbox for all server objects selected in the viewport.
	Insert page reference	Inserts a page reference symbol pair on the selected links for which origin and destination servers are on different pages . Requires a completed Page setup procedure.
	Remove page reference	Removes the selected page reference symbol pairs. Requires a completed Page setup procedure.
	Remove empty pages	Removes all pages that do not contain any server or link objects. Requires a completed Page setup procedure.

Tools menu

	Configuration	Opens the Configuration window, as will double-clicking the viewport.
	Pictures	Opens the Picture management console.
	Layers	Activates the Layer control panel.
	Page background editor	Opens the background editor window. Requires a completed Page setup procedure.
	Preferences	Opens the Preferences utility to change preference settings of the application and the project.

Window menu

	Background	Changes the background color of the viewport.
	Arrange windows	Opens the Window arrangement utility.
	Windows list	Presents a list of all open window titles. Clicking on an item in the list activates the window.

Help menu

	Getting started	Opens the Getting started section of the help document.
	Renque help	Opens the help document.
	Renque website	Opens the Renque website in the default internet browser.
	Check for updates	Connects to the internet and checks if a more recent Renque release is available for download.
	About	Displays the application's introductory splash window. Click the splash window to close it.

Toolbar only commands

	Pan	Toggles left mouse button viewport manipulation.
	Zoom	Toggles the zoom function for viewport manipulation.

2.5. Simulation toolbar

The simulation control toolbar is located underneath the viewport. The main simulation run commands **Reset**, **Restart**, **Pause** and **Animate** of the Run menu are also available as control button on the simulation bar, together with some additional controls and simulation state indicators.

Simulation run buttons

The **Reset** button resets the current simulation run and discards all simulation data. The **Restart** button starts a new simulation run. The **Pause** button toggles between the paused and running simulation state.

Animation on/off

The Animation button **t**oggles animation on and off.

Animation speed

The Animation speed slider controls the animation speed. Moving the indicator to the left increases the animation speed. The speed range is adjustable by the Speed control range extent item on the configuration window.

Simulation preview

The **Simulation preview** button toggles the Simulation preview display mode. If a simulation is running, the display style of objects on the viewport is determined by the Server simulation display options and Link simulation display options. Simulation preview applies the simulation display style to the viewport when the simulation is reset.

Simulation state pane

The Simulation state pane is the text field adjacent to the preview button on the Simulation toolbar. It reports the sequence number and state of the current replication. The text in this panel colors red if the simulation has a runtime error.

Simulation time panel

The Simulation time panel is the rightmost control on the Simulation control toolbar. The simulation time panel reports the current simulation time in a user-specified format. A mouse click on this panel activates a text edit field for the time display format string using text markup. escape sequences for clock properties.

2.6. Status bar

The Status bar is located at the bottom edge of the window and is divided into panels, as indicated in the figure.

Layer pane

The Layer pane (Item 1) displays the name of the Active layer. Clicking the layer panel activates the Layer control panel.

Mouse location panel

The mouse location panel (item 2) displays the mouse x/y-coordinates on the viewport

Viewport options panel.

The viewport options panel (item 3) contains two toggle buttons. The **Grid** button toggles the display of the Page grid. The **BKGND** button toggles the display of the viewport Page background.

Object snap panel

The object snap panel (item 4) contains a set of four toggle buttons, which aid precise positioning of objects on the viewport. The **Grid snap** button enables snapping to the viewport page grid. The **Object snap** button enables snapping of server edges end link end points. The **Ortho button** results into preservation of vertical or horizontal coordinates of displaced viewport objects. The **Drag threshold** button implements a threshold distance for mouse drag operations, which can be used to prevent unintentional displacement of objects in the viewport by small mouse movements. The threshold also applies to viewport translation by mouse dragging.

2.7. Layers

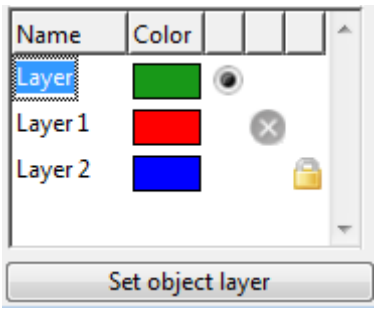
The server and link objects on the viewport are organized in layers. The layers have a specified color, and may be **locked** and **hidden**. The main purpose of layers is to cluster objects that are related in some meaningful way, for example by a shared function in the model. The organization of a project into layers offers many advantages to the project designer. The ability to hide layers and to apply different colors to layers makes it easier to visually distinguish objects in the viewport. Objects in locked layers cannot be selected in the viewport.

Active layer

New objects created on the viewport are placed in the layer that has been assigned **active layer**. The active or current layer can be identified in the Status bar and by the color of the default templates in the Toolboxes.

Layer control panel

The layers defined within project are managed by the Layer control panel, which is activated by a mouse click on the left-most panel of the Status bar or by the **Layers** command of the Tools menu. The Layer control panel is a pop-up control that presents a list of all layers defined for the project. It enables the creation and deletion of layers, modification of layer properties, and assignment of the current layer. It is also used to place objects on the viewport in a different layer.



The column labeled ***Name*** displays the layer name. The layer name can be edited in-cell, by double clicking the name field or by pressing a key on the keyboard. The assigned name must be unique and has the same format requirements as the Name property of servers, links and components. The column labeled **Color** determines the default display color for objects contained in the layer. A click on the Color column activates a color selection dialog. The radio button in the third column identifies the **Active layer** of the viewport. The two other columns present additional layer properties indicated by an icon. The cross-out icon signifies that the layer is **hidden**. The padlock icon signifies that the layer is **locked**. A mouse-click on either of the two leftmost columns toggles the corresponding layer property.

The **Set object layer** button appears on the Status bar when the Layer list is activated. By pressing this button, all selected server and link objects on the viewport are placed in the selected layer in the Layer list. New layers are created by the **Add layer** command of the contextual menu of the layer list. The **Del key** deletes the selected layers in the Layer list.

While the layer panel is visible, the window's Menu bar & Toolbar are disabled, as are all other windows of the application. The Layer tool is deactivated by clicking on the **viewport**, or pressing the **Escape key** on the keyboard.

2.8. Blocks and fusions

Blocks and fusion are named object sets that act as a single object. You can use them to create composed templates and compress specific parts of a model. They can help you save time and make your model easier to view. Block and Fusion servers can be stored as server templates in the Toolbox.

Blocks

A set of server and link objects selected in the viewport can be joined into a block server. A block server represents the block members collectively by a single server in the viewport. The constituents of the block server cannot be individually selected, rearranged, edited or modified. Links can still be connected to the individual servers contained in a block.

The **Create block** command of the Blocks Format menu merges all selected objects on the viewport, including block servers, into a new block server. The **Explode block** command reverses this transformation for all selected block servers on the viewport.

Fusions

The **Fuse blocks** command of the Blocks Format menu transforms the selected block servers in the viewport into fused block servers. Fused block servers are useful to represent a section of a project by a single server object. The default image of a fusion server is slightly different from the default server image.

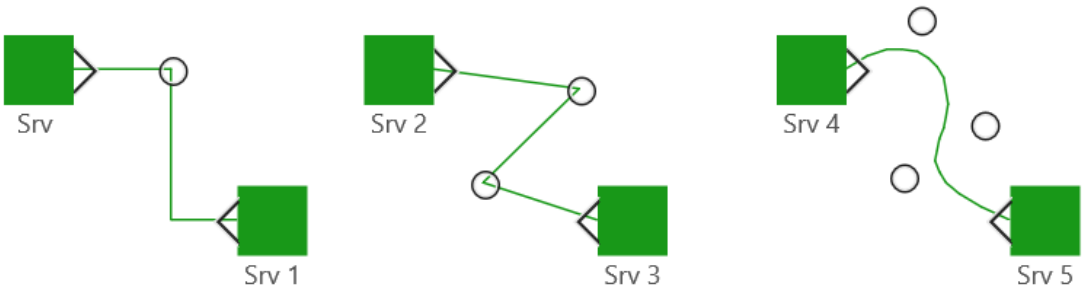
A link that is connected to a fused server, but not fused itself, is shown as connected to the fusion server in the viewport. Links can be connected to a predetermined destination or origin server contained in a fusion. The **Unfuse blocks** command transforms all selected fused block servers back to a regular block server.

Fusion window

A fused block server may be displayed in a separate Fusion window. The **Open fusion window** command of the Blocks Format menu opens new fusion windows for all selected block and fusion servers in the viewport. The fusion window is identical to the Workspace window, but does not have the File menu and some items in the Help menu. The fusion window permits modification of the block server by the same methods used in the viewport of the workspace window and supports all simulation Simulation display features. Changes made to the Layer control panel and the simulation control toolbar of the workspace window and fusion windows apply to all open windows. The **Close block window** command closes the block windows for all selected block servers in the viewport.

2.9. Link curve types

The shape of a link curves in the viewport is determined by the curve format type and a series of control points. The application supports the curve types *Orthogonal, Polygon* and *Spline*.



Orthogonal **Polygon** **Spline**

The curve types exist for display purposes only: they have no logical distinctions in a model.

2.9.1. Link curve type selection

The curve type of a link is determined by the **Curve type** item in the Format Menu. This control is available both for new links and for object editing.

2.10. Viewport pages

Completion of the **Page setup** dialog of the File menu divides the viewport into pages. The specified page properties may be reviewed by the Page layout item of the Preferences console

Page grid

The Grid size item of the Preferences console allows definition of a page grid as reference for the arrangement of objects on a page. The page grid is only shown on viewport pages that have at least one server or link object. The grid can be hidden by the **Grid** button on the Viewport options panel.

Drawings

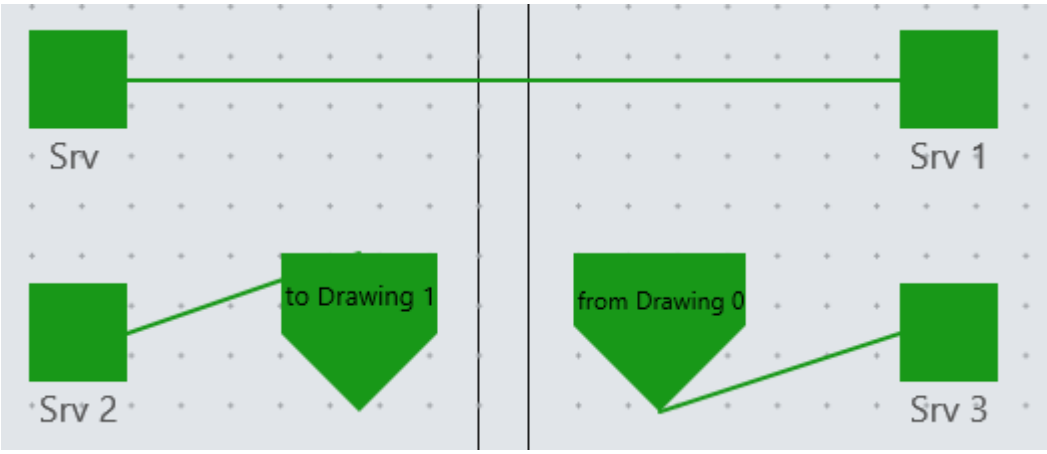
Each page on the viewport represents a drawing. The properties of a drawing are specified on the Specs tab of the Model console of the configuration window.

Page background

The **Page background editor** item on the Tools menu opens the background editor. The page background is drafted in the viewport of the background editor tool similar to the simulation model in the Workspace window, but cannot include link objects. The page background is displayed on every page of the workspace viewport. The properties of individual drawings can be expressed by text markup for server objects in the background. The page background can be hidden by the **BKGND** button on the Viewport options panel. Note that objects placed on the page background are not part of the simulation model.

Page reference servers

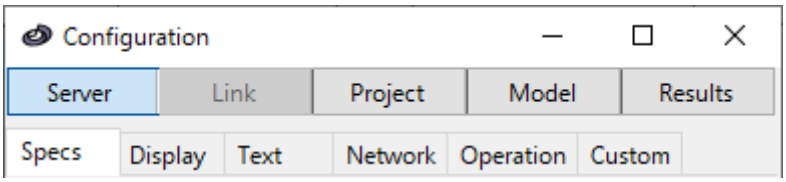
A page reference server pair presents a method to systematically arrange page crossing links. A link is transformed into a page referencing link by the **Insert page reference** item of the Transform menu. This transformation spits the link into two parts on the viewport, each connected to a newly created page reference server. The page transition is depicted by placing each page reference server on the matching source or destination page of the original link, which will consequently be represented on both pages by either of the two split parts.



The page reference servers have no logical function in the simulation and the two link parts both represent the transformed link in the simulation. Simulation display is performed on both link elements. The page location of the page reference servers can be expressed by text markup. The page reference construction is removed from a link by the **Remove page reference** item of the Transform menu.

3. Object configuration

The Configuration window provides access to the properties of objects on the viewport and many other elements of the project. The window is opened by the **Configuration** menu on the Tools menu or by double-clicking the viewport.



The category bar at the top of window selects the main category for which properties are displayed. Selecting a category displays the corresponding console with all relevant control items.

The properties configured by the console categories are:

Server	Server object properties
Link	Link object properties
Project	Properties of drawing objects or of the project itself
Model	General properties and properties of objects that are not represented on the viewport
Results	Simulation results and related information

The tab strip beneath the category bar generally selects a group of related properties. The composition of this tab strip can vary between the different consoles.

This chapter discusses property configuration on the **Server**, **Link** and **Project** consoles.

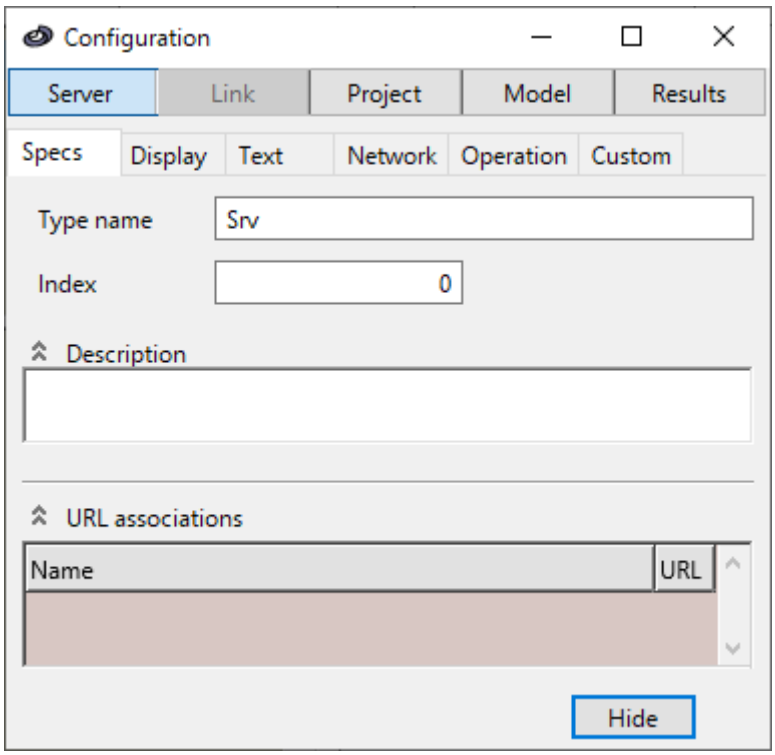
Viewport context

The configuration window can be opened from the Workspace window and from a Fusion window. The window from which the configuration window is opened represents the viewport context. The Server, Link and Project consoles only display and configure properties of objects that have the selected state in the viewport. For the Project console, Drawings are considered to be selected if one or more objects on the drawing have been selected. If no drawings are selected, the Project console references the current project.

If the selected objects do not have a uniform value for a specific property, check boxes are grayed out, and pop menus are left empty, whereas in text fields and combo boxes the keyword <*various*> is displayed. In general, a control item on the console is disabled or hidden if it configures a property that does not affect any of the selected objects.

3.1. Specs

The **Specs tab** on the Server, Link and Project consoles presents general properties of selected servers, links and drawings. If no drawings are selected, the Project console presents the properties of the project.



Type name

The Type name property may contain a maximum of 255 characters in the ranges A-Z, a-z, 0-9 and the underscore _ character. The property must start with an alphabetic character and is case-insensitive. The name of an object consists of type name followed by the non-zero value of the below-mentioned Index property. When the Type name property is altered, the Index property may change for any of the selected objects to render the name property unique in the project.

Index

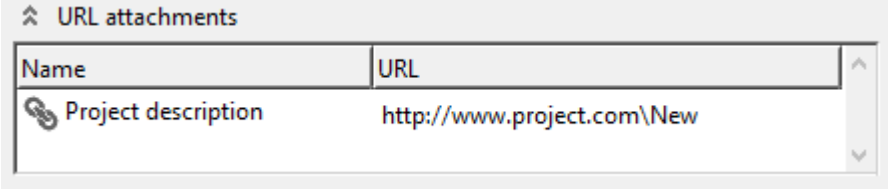
The index property of an object in an integer number >0. The name of an object consists of the above-mentioned type name followed by the non-zero index value. The name of each object is unique in the project. When the index property is changed, indexing of the selected objects starts with the specified value, where the index value of non-selected objects with the same type name are skipped. The default tag property of an object is the type name followed by a space and the non-zero index.

Description

Descriptive text for the item.

URL attachments

Displays a list of attached URLs to the selected viewport object, drawing or the project.



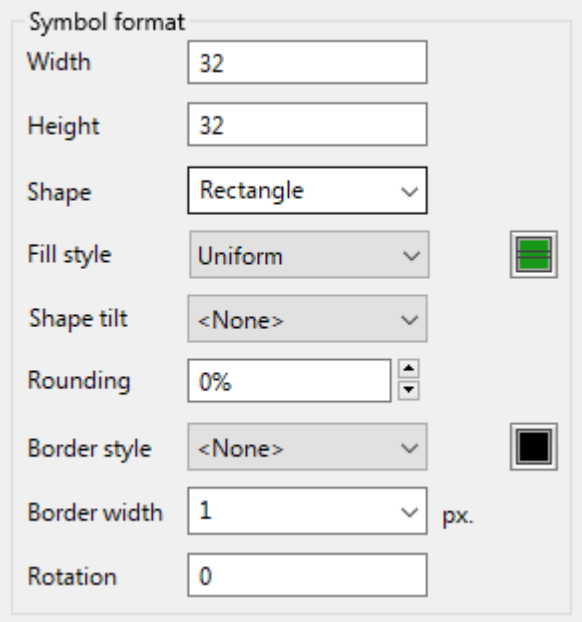
The **Add item** command of the contextual menu of the list adds a new untitled blank URL attachment to the list. The **Edit name** contextual menu command activates the edit mode of selected cell in the name column, as does pressing the space bar. The **Edit URL** contextual menu command activates the edit mode of selected cell in the URL column. The **Browse file** command allows opens a file selection dialog for assignment of a file URL to the selected item in the list. The **Show in folder** command opens a file URL in the system file browse. A **double-click** on the list opens the URL of the selected item with the appropriate system application.

3.2. Display

The **Display tab** on the Server and Link consoles presents the display properties of servers and link objects.

3.2.1. Symbol format

A server object is represented on the viewport by a constructed symbol, consisting of a base shape and several other elements. The base shape can be an ellipse, a polygon or a chart. The shape can be solid or transparent, can be rounded and rotated and may have a border. The symbol can also have a curve, a picture and text displayed on top of the base shape. All symbol properties are controlled of the Display tab, except for the text properties, which are controlled on the Text tab.



Width

Determines the width of the server in the viewport. The value is specified in the value selected for the Project base unit. The command **Make symbols equilateral** of the right-click activated contextual menu of the text field adjusts the Width property of symbol servers to obtain an aspect ratio that renders polygons equilateral and ellipses circular.

Height

Determines the height of the server in the viewport. The value is specified in the value selected for the Project base unit. The command **Make symbols equilateral** of the right-click activated contextual menu of the text field adjusts the Height property of symbol servers to obtain an aspect ratio that renders polygons equilateral and ellipses circular.

Shape

Determines the base shape of the symbol. Selection of the *Chart* value furnishes the symbol with an additional element to display numerical data in a chart diagram. Pressing the adjacent toggle button **Config** enables configuration of the chart properties as described in the Chart Configuration section.

Fill style
Determines the fill style of the base shape. A click on the adjacent button opens a color section dialog that allows selection of the fill color. For the two Gradient values a second color button appears which represents the secondary gradient color.

Shape tilt
Tilts the symbol shape by 90-degree angle increments.

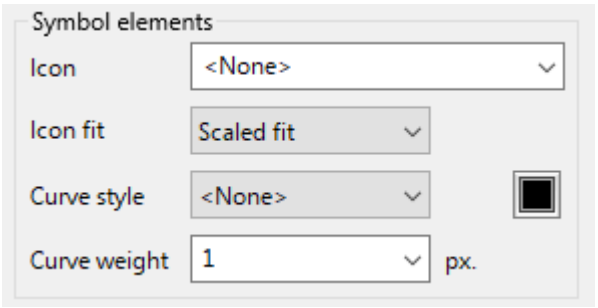
Rounding
Determines the corner rounding of the border within a range of 0 - 100%

Border style
Determines the border line style. A click on the adjacent button opens a color section dialog that allows selection of the border color.

Border Width
Determines the border width specified in mm.

Rotation
Determines the clockwise rotation angle of the symbol in degrees.

3.2.2. Symbol elements



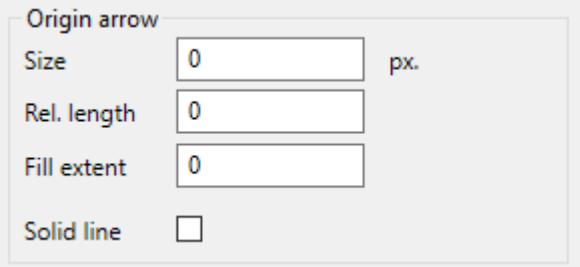
Icon
Determines the picture object to be displayed as icon in the server image. Type the name of the desired picture object or select it from the combo box.

Icon fit
Determines the arrangement of the above-mentioned icon in the symbol. Possible values are, *Center* (centered without size change), Scaled fit (scaled to fit the symbol with preservation of aspect ratio), *Filled fit* (stretched to fill the symbol), *Tile* (Repetitive tiling from top-left corner, cropped at left and bottom edges), *Shrink* (c*enter* and if larger than the symbol: *Filled fit*)

Curve style
Determines the curve line style. A click on the adjacent button opens a color section dialog that allows selection of the curve color.

Curve weight
Determines the curve width specified in the value selected for the Project base unit. The value *Hairline* represents the smallest value for the resolution of the selected printer.

3.2.1. Link arrow properties



Arrow size
Determines the distance of the arrow hand tip to the edge of the link curve, specified in mm or pixels.

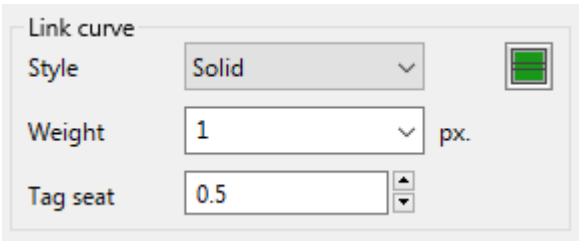
Rel. length
Determines the length of the arrow as fraction of the above-mentioned size property. The default value of 1.732 for the standard link templates is consistent with an equilateral triangular arrow head.

Fill extent
Determines the length of the solid arrow area along the curve as fraction of the above-mentioned relative length property

Solid line
Forces a solid line for the arrow outline for dotted and dashed curve styles

Serial
Determines whether or not arrow display is repeated for every curve segment of a segmented curve.

3.2.2. Link curve



Style
Determines the curve line style. A click on the adjacent button opens a color section dialog that allows selection of the curve color.

Weight
Determines the curve width specified in mm. The value *Hairline* represents the smallest value for the resolution of the selected printer.

Tag seat
Determines the tag location at the link curve in the range 0 to 1, where 1 denotes the destination location.

3.3. History

The **History tab** on the Project console presents revision information of the project and of the drawings in the project.

Revision

Displays a list of revision numbers of the selected drawings in the project and of the project itself. A drawing is selected if any of the viewport objects in the drawing are selected.

The revision number is displayed in bold if a revision increment is pending because of modifications made to the drawing or the model since the last revision increment. Drawings obtain a pending revision increment if the drawing display on the worksheet changes or any of the drawing properties are altered. The project obtains a pending revision increment for any change made to the simulation model.

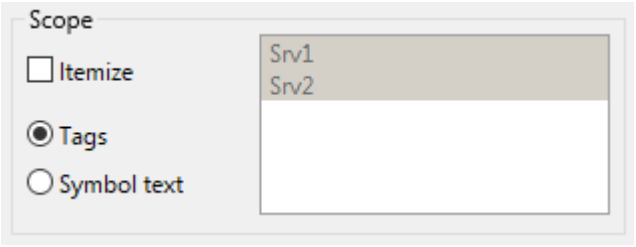
The revision numbering format and updating method are controlled by the revision control item in the project preferences. For manual revision control, the lowest level revision number may be incremented manually by the **Increment** command of the contextual menu of the list. This command is effective only if a revision increment is pending. The **Increment level** submenus of the contextual menu allow increments of higher-level revision numbers. This action resets all lower revision numbers and removes pending increments. In case of a drawing revision change, the model revision is adjusted to match the largest revision number of the drawings

3.4. Text

The **Text tab** on the Server and Link consoles presents the caption properties of servers and link objects in the viewport.

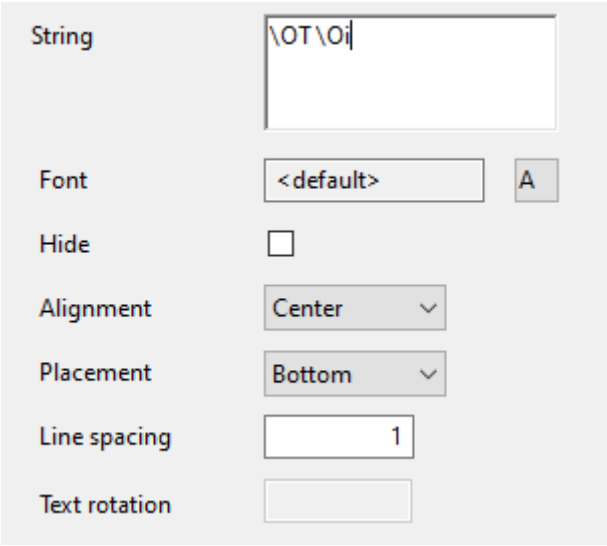
3.4.1. Scope

The scope frame determines which properties are referenced by the other controls on the tab. Selecting the **Symbol text** option (not available for links) changes the scope from server and link tags to the text displayed in symbol servers.



Checking the **Itemize** box allows exclusion of individual objects by deselection in the list on the right, which presents a list of all server of link objects selected in the Viewport context of the configuration window.

The other controls on the Text tab manage the text string to be displayed and the display style.



String

Determines the text displayed. Multiple lines are allowed. Text markup can be used to insert object properties in placeholders. Symbol text also recognizes clock escape sequences. Any combination of escape sequences can be used. All characters that do not represent an escape sequence are displayed as literal text. The script command `Text` is used to access symbol text at runtime.

Font

A click on the Font text field opens a **Font selection dialog**, which allows defining the text font properties. Note that for symbol text the font size is specified as percentage of the extent to fit the text region in the symbol. Clicking the adjacent

color selection button opens a dialog for selection of the text color. A click on the color button while pressing the **Ctrl-key** assigns the layer color to the text color. A click on the color button while pressing the **Shift-key** assigns the viewport foreground color to the text color.

Hide
Hides the text in the viewport

Alignment
Determines the horizontal text alignment. Possible values are *Left*, *Center* and *Right.* For tags only effective if the tag has multiple lines.

Placement
Determines the vertical text alignment. Possible values are *Top*, *Middle* and *Bottom.*

Line spacing
Determines the line spacing of the text display in the viewport, as fraction of the normal line spacing.

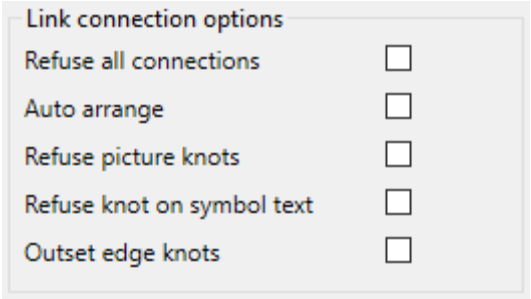
Text rotation
Determines the clockwise rotation angle of the link tag in degrees. An empty value of the text filed denotes alignment of the tag with the curve at the Tag seat location. Not available for servers.

3.5. Network

The **Network tab** on the Server and Link consoles configures the connection properties of servers and link objects in the viewport.

3.5.1. Link connection options

The link connection options frame presents several server options that configure the manner by which new links are connected to the server.



Refuse all connections
If checked, links cannot be connected to the server.

Auto arrange
Determines the method by which link connections are organized. If checked, new link connections are snapped to the server edges in a uniformly spaced arrangement. If unchecked new links can be connected to any location on the server. Changing the value of the Auto arrange property does not affect existing link connections

Refuse picture knots
If checked, link connections cannot be snapped to picture knots of the symbol icon.

Refuse knot on symbol text
If checked, link connections cannot be snapped to the outline of symbol text.

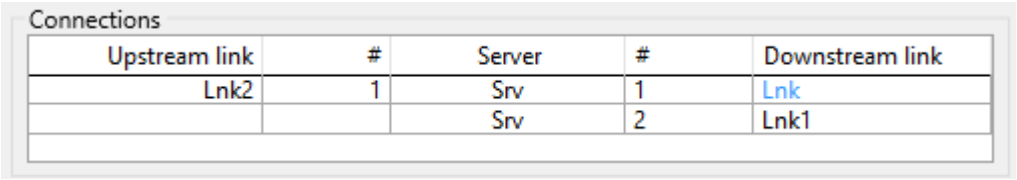
Outset edge knots
If checked, links connections are shifted away from the server edge by half the curve width, such that the connected link curves do not overlap with the server image.

3.5.2. Auto arrangement options

Determines the arrangement of link connections under application of the above-mentioned Auto arrange option. The control has three properties, organized in columns for the four sides of the server image as indicated in the first column. The **Max. knots** column determines the maximum number of permitted connections for each side. The value can be set by clicking the corresponding cell and typing the desired number. An empty field denotes that there is no limit to the number of connections. The check boxes in the **Static** column denote that the link connection spacing is predetermined in accordance with the specified Maximum knots value, independent of the number of connections. If checked, links can only be connected to a free knot position on the edge. The **Spacing** column determines the spread of the connections on the edge. A value between 0 and 1 can be specified, which denotes the relative length of connected section of the edge.

3.5.3. Connections

The connections frame contains a list of all connections of the selected server or link objects.



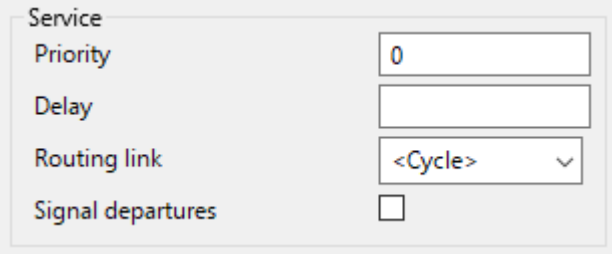
The list has 5 columns. The center column shows the name of the selected object. The two outer columns display the names of the connected origin and destination servers for links or the names of the connected links on both sides for servers. The two columns on either side of the center column labeled with a hashtag display the sequence position of the connection. The names of selected objects are displayed in the system's selection highlight color in the two outer columns. For links, the destination and origin node can be changed by selecting another server name from the drop-down list that appears upon a mouse click on the popup-arrow at the edge of the list.

3.6. Operation

The **Operation tab** on the Server and Link consoles configures the behavior of server and link objects in the simulation, including some animation properties.

3.6.1. Service

The **Service** frame displays server properties that determine the non-customized behavior in a simulation. The frame displays the initial reset-state value of these properties.



Priority

Determines the Priority value of calendar events generated by the server. Permitted values are integer numbers in the range -32765 to +32765. A value assigned by the `Priority` script property is indicated between parenthesis following the initial reset-state value.

Delay

Determines the entity storage time for routing by the `ArrivalAccept` procedure of the server. The Delay property is set to *Nil* if the text field is left empty, which causes the entity to be stored for an unrestricted period of time. The component types attribute, variable and distribution can be assigned by entering the component name. The simulation evaluates and applies the current value of the assigned component when the Delay property comes into effect. A warning is created in the runtime error list if the evaluation yields an invalid timing property. In that case the value *Nil* is applied. Permitted numerical values are floating-point numbers >= 0.

Routing link

Determines the method for selection of a downstream recipient link for entity routing by the `ArrivalAccept` procedure of the server. The default value <*Cycle*> results into entities being sent in a cyclic pattern to the downstream links in top-down connection order. The value <*Random*> results into random selection of a link, whereas the routed entity is destructed for the value <*None*>. Entering a positive integer number in the text field results into selection of the corresponding link in top-down connection order. Small integers can be selected from the drop-down menu. Larger numbers must be assigned by entering a number in the text field. The entity is destructed if server has no downstream links and also if the assigned number is larger than the number of downstream connected links, in which case a warning is created in the runtime error list.

Signal departures

Option that causes a departure signal to be sent to the server before a routed entity leaves the server by execution of a routing or expiration event.

3.6.2. Data options

The data frame has several options controlling how simulation data is handled for the server or link object. Changes made to these properties take effect at the next simulation restart.

Collect statistics
Activates or deactivates statistics collection for the object.

Record data
Activates or deactivates data recording for the object.

3.6.3. Server simulation display options

Hide: residents; curve; resident count; tag; icon
Hides the indicated items during a simulation run.

Animate retention
Enables animation of the residence progress of server residents.

Show first curve non-fit
Resident icons are displayed on the server curve by default only if the icon fits completely on the curve. If applied, the first resident icon is displayed even if it doesn't fit on the curve.

3.6.4. Link simulation display options

Hide: residents; curve; travel
Hides the indicated items during a simulation run.

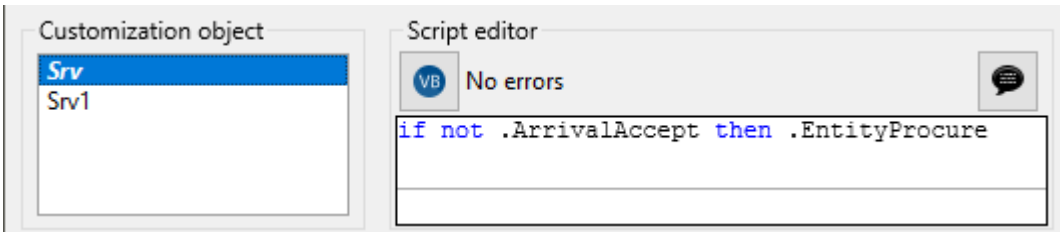
Show first link curve non-fit
Resident icons are displayed on the link curve by default only if the icon fits completely on the curve. This option imposes display of the first resident icon even if it doesn't fit on the curve.

3.7. Custom

The **Custom tab** on the Server and Link consoles allows modification of the behavior of a server or link object in the simulation by customization. A customization changes the behavior of a server or link by execution of a script in response to a signal invoked by specific incidents in the simulation.

3.7.1. Customization control

The customization object is selected in the Customization object list. Customization scripts for the customization object are composed in the Script editor frame.



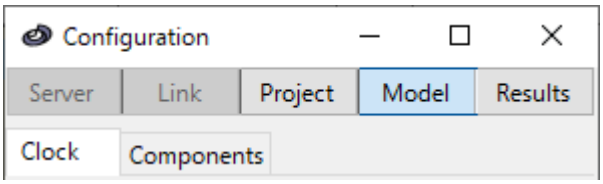
The **Script editor** frame displays the customization script of the objects selected in the **Customization object** list. The text <*various*> appears in the text area if the script code varies between the objects selected in the list. In that case the contextual menu command **Replace with new script** can be used to create a new uniform script for all selected server objects.

The compiler button at the top left of the frame indicates the script language selected for the present customization with an icon. The script language selection may be altered by clicking on this button with the right mouse button. Pressing the button precompiles the script code, after which any syntax errors will be highlighted in the text area. Additional instructions for operation of the script editor are given in the topic Script editor.

Other scripting topics
Script coding techniques are discussed in detail in the Customization scripts topic.

4. Model configuration

The configuration window provides access to general properties of the model and many other elements of the project. The window is opened by the **Configuration** menu on the Tools menu or by double-clicking the viewport.



The category bar at the top of window selects the main category for which properties are displayed. Selecting a category displays the corresponding console with all relevant control items.

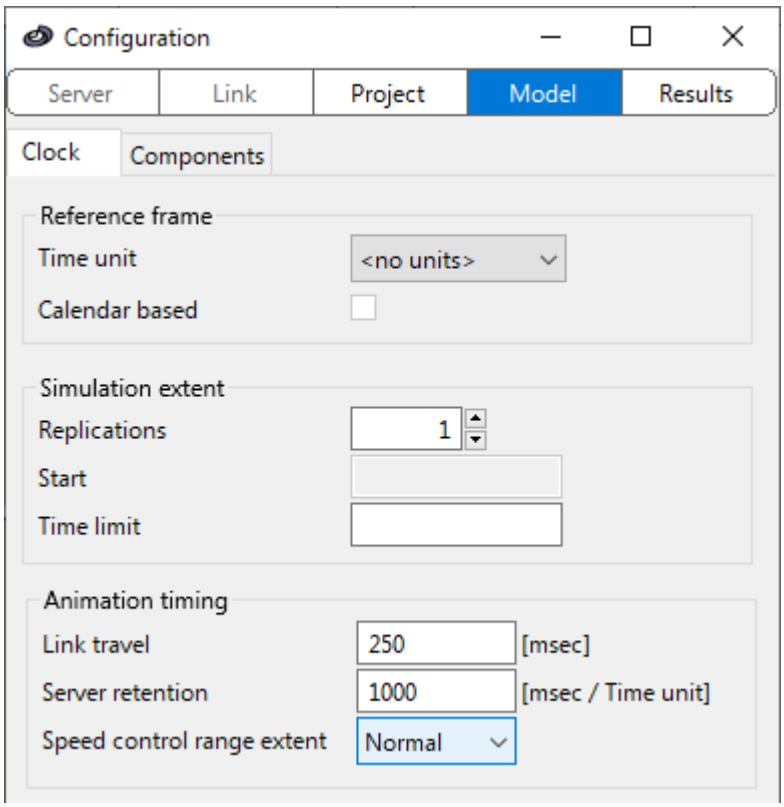
The **Model** console contains general properties of the model that are not associated with objects represented on the viewport.

This chapter discusses model property configuration, organized by the tabs on the tab strip beneath the category bar:

- **Clock** Simulation clock properties
- **Components** Simulation objectsa not represented on the viewport

4.1. Clock

The **Clock** tab of the Model console determines the time span of the simulation and the properties of a user-adapted built-in clock, which controls the Simulation timing.



4.1.1. Reference frame

Time unit

Defines the physical time unit applied to the clock. The Clock program frame is shown if a value other than *<No Units>* is selected.

Calendar based

Selects the Gregorian calendar as reference frame for the simulation clock. If checked, the Timetable calendar is enabled. The check box is not available if the *Dimensionless* value has been selected in the above-mentioned Time unit dropdown list.

4.1.2. Simulation extent

Replications

Sets the number of replications for a simulation run. The maximum number of replications is 32000.

Start

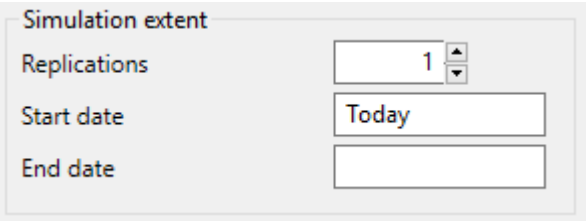
Specifies the clock start weekday. Not available if the *Dimensionless* value has been selected for the Time unit of the clock. The Start property requires selection of a weekday from the dropdown menu.

Time limit

Specifies the maximum run time for all simulation replications in the non-Calendar based clock. The run time is expressed in the selected Time unit as

positive floating-point number. An empty text field (Default) signifies that the simulation is performed endlessly.

For the Calendar-based clock the simulation start and end are determined by calendar dates.



Start date
Specifies the start date for the clock calendar. The property requires specification of a date in the date format of the operating system. The default value *Today* denotes the current date of the operating system clock.

End date
Specifies the end date for the clock calendar. The property requires specification of a date in the data format of the operating system. The value *Today* denotes the current date of the operating system clock. An empty text field (Default) signifies that the simulation will be performed endlessly.

4.1.3. Animation timing

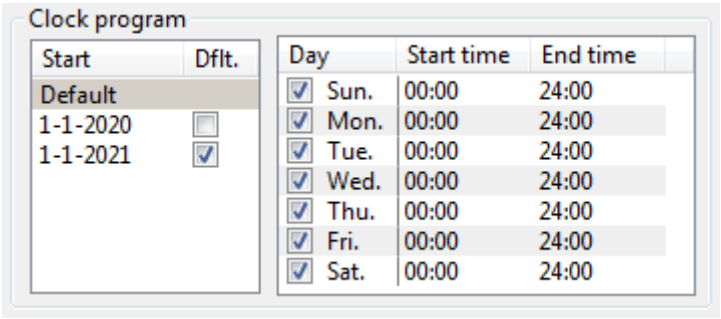
Link travel
Specifies the display time for travel animation in milliseconds real time for the Animation speed slider in the middle position. The text field accepts positive integer numbers between 1 and 60000.

Server retention
Specifies the display time for retention animation in milliseconds real time per simulation Time unit for the Animation speed slider in the middle position. The text field accepts any positive floating-point number.

Speed control range extent
Specifies the span of the of the animation speed range controlled by the Animation speed slider. The pulldown menu requires a selection out of five values.

4.1.4. Clock program

The clock time is expressed in the 12 or 24-hour clock system and weekdays are distinguished if a dimensional Time unit has been selected for the clock. The dimensional selection also presents the **Clock program** frame. The Calendar-based clock program is an agenda of user-defined timetables which are consecutively activated on a specified start date. A timetable defines the operative time pattern of each week day. The **Timetable calendar** control located on the left of the frame contains a list of clock program timetables organized by start date. The **Timesheet** control located on the right displays the configuration of the selected timetable item in the timetable calendar. If the clock is not Calendar-based the *Default* item is the only timetable defined and the timetable calendar control is disabled.



Timetable calendar

The timetable calendar control contains a number of timetables organized in rows. The start date of an item is specified in the column labeled *Start* by typing a date value in the system's date format. Timetables can be added and deleted by commands on the contextual menu of the agenda. The first row with start value *Default* is fixed and cannot be removed. A timetable is valid if it has a start date later than the start date of the previous valid timetable. Invalid timetables are ignored in the simulation. A timetable becomes active on the start date value and remains active until the start date of the next valid item in the calendar. A checked box in the column labeled *Dflt* denotes that the timetable item shares the timesheet of the default timetable item of the first row, instead of having a unique customized timesheet.

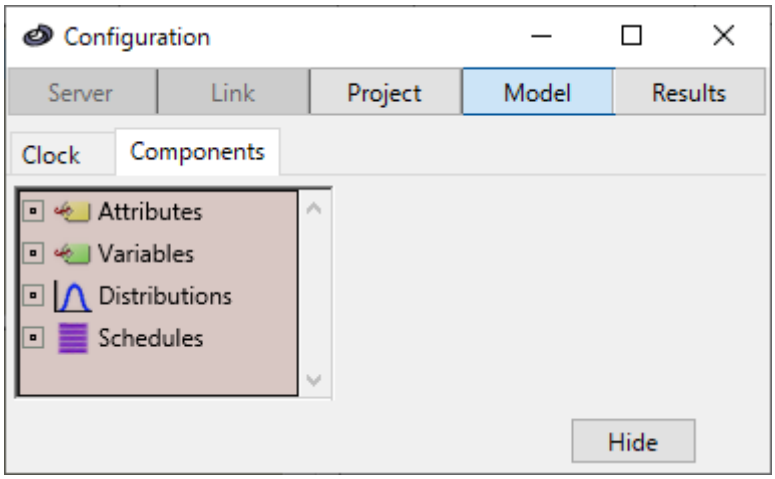
Timesheet

The Timesheet control displays the properties of the selected timetable in the above-mentioned timetable calendar. The timesheet has a row for every day of the week. The first column specifies the abbreviated weekday name**,** with a check box to enable/disable the corresponding day in the timetable. The other columns define the **Start time** and **End time** of day segments in column pairs. By default, the timesheet has a single column pair. Column pairs can be added using the **contextual menu** of the timesheet and of an active cell. A day segment is invalid if the start time value is later than the end time value, or if the start time value is earlier than the end time value of the previous valid day segment. A week day with an invalid first day segment is skipped in the simulation. If the **Uniform week** item of the contextual menu of the timesheet is checked, the selected timetable is uniform for all days of the week and the timesheet has only one row.

If a timesheet without any valid day segments, and thus of zero duration, is activated, the timetable calendar immediately jumps to the start date of the next valid item in the Timetable calendar that has a non-zero duration timesheet.

4.2. Components

The **Components** tab on the Model console provides access to component objects. Components are user-creatable project elements that are not displayed on the viewport. The Component repository on the left side of the window contains a categorized list of the component objects defined within the project. The repository is used to create, delete, and select components. New components are created with the **New** and **Duplicate** commands of the contextual menu of the Component repository. Components are removed be the **Delete** command.

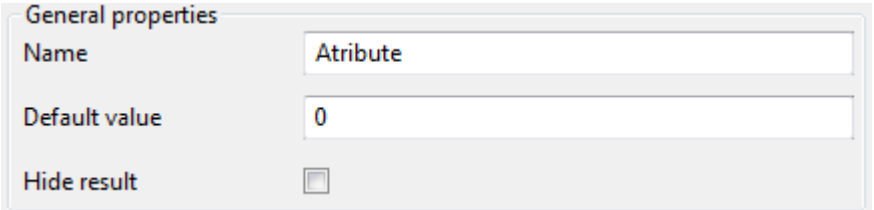


Component properties are displayed in frames, which are only visible if all components selected in the component repository are of the same type.

4.2.1. Variables and attributes

Variable and Attribute components are used to store data. Attributes are used to store data in entities. Variables are used to store data either with a universal scope, or in a particular server or link object.

General properties



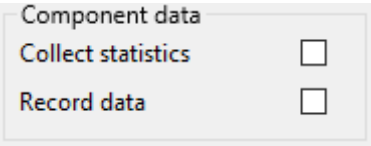
Name

Name of the component. Selected items may be automatically renamed by altering the numeric suffix of the assigned name property in order to render the name of each object unique in the project.

Default value
Default value of the component. Permitted values for the default property are integer and floating-point numbers, strings and booleans. The format of the specified value is recognized automatically.

Hide result
Hides the component in the Variable value data sheet or the Attribute value data sheet.

Component data
The component data frame has several options controlling how simulation data is handled for the component object. Changes made to these properties take effect at the next simulation restart.

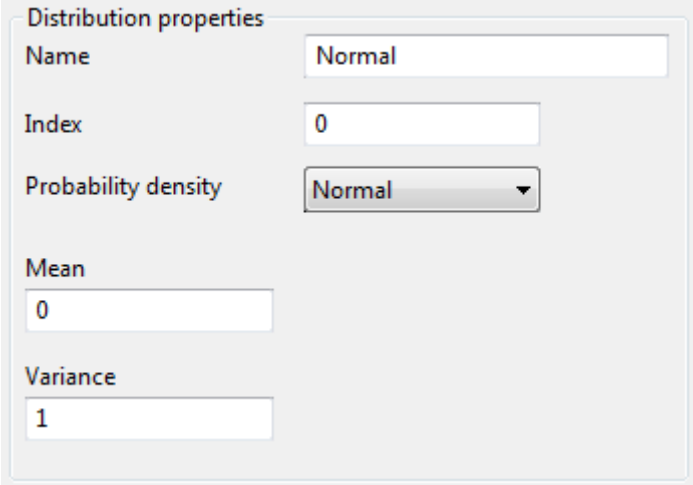


Collect statistics
Activates or deactivates statistics collection for the component.

Record data
Activates or deactivates data recording for the component.

4.2.2. Distributions

Distribution components are objects that generate a random number when evaluated. Distribution components are defined by a probability density function (PDF), which determines the spread of the numbers generated by the component.



Type name
The Type name property may contain a maximum of 255 characters in the ranges A-Z, a-z, 0-9 and the underscore _ character. The property must start with an alphabetic character and is case-insensitive. The **name** of an object consists of the type name followed by the below-mentioned non-zero index value. The name of each object is unique in the project.

Index

The index property of an object in an integer number >0. When the index property is changed, indexing of the selected objects starts with the specified value, where the index value of non-selected objects with the same type name are skipped.

Probability density

Selects the probability density function for the distribution component. The available functions are given in the table below. In this table, the parameter x indicates the random value. Where not indicated otherwise, x is a real number without range limits. The symbol Γ represents the Gamma function.

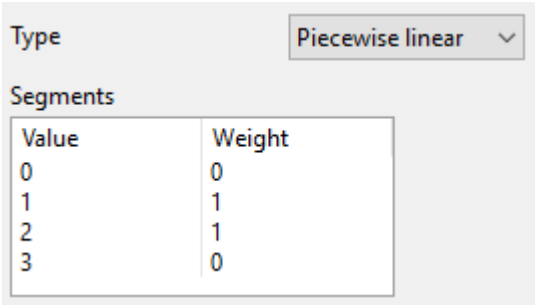
Name	*Parameters*	*Probability density function*	*Domain*
Beta	Shape a Shape b	$\frac{\Gamma(a+b)}{\Gamma(a)\cdot\Gamma(b)}\cdot(1-x)^{b-1}\cdot x^{a-1}$	a > 0, b > 0
Binomial	Trials N Probability p	$\frac{N!}{x!\cdot(N-x)!}\cdot p^{x}\cdot(1-p)^{N-x}$	0 <= p <= 1 N = 0,1,2 ; x = 0,1,2..N
Exponential	Mean λ	$\lambda\cdot\exp(-\lambda\cdot x)$	λ >= 0 , x >= 0
Gamma	Shape k Scale θ	$\frac{x^{k-1}\cdot\exp\left(\frac{-x}{\theta}\right)}{\theta^{k}\cdot\Gamma(k)}$	k > 0, θ > 0, x>= 0
Normal	Mean μ StdDev σ	$\frac{1}{\sigma\cdot\sqrt{2\cdot\pi}}\cdot\exp\left[\frac{-(x-\mu)^2}{2\cdot\sigma^2}\right]$	σ > 0
Poisson	Shape ν	$\frac{e^{-\nu}\cdot\nu^{x}}{x!}$	v >= 0 x = 0,1,2...
Triangular	Minimum a Modus m Maximum b	$\frac{2\cdot(x-a)}{(b-a)\cdot(m-a)}$	a <= x <= m
		$\frac{2\cdot(b-x)}{(b-a)\cdot(b-m)}$	m < x <= b
Uniform	Minimum a Maximum b	$\frac{1}{b-a}$	a <= x <= b
Weibull	Shape α Scale β	$\alpha\cdot\beta^{-\alpha}\cdot x^{\alpha-1}\cdot\exp\left[-\left(\frac{x}{\beta}\right)^{\alpha}\right]$	α > 0, β > 0 x >= 0
Piecewise	Segment end point table	Defined by a series of continuous linear segments	

Distribution parameters

Parameter data may by entered in the text fields located below the PDF combo box. The available parameters depend on the selected distribution function. The

parameter value ranges allowed are listed in the right-most column of the table above.

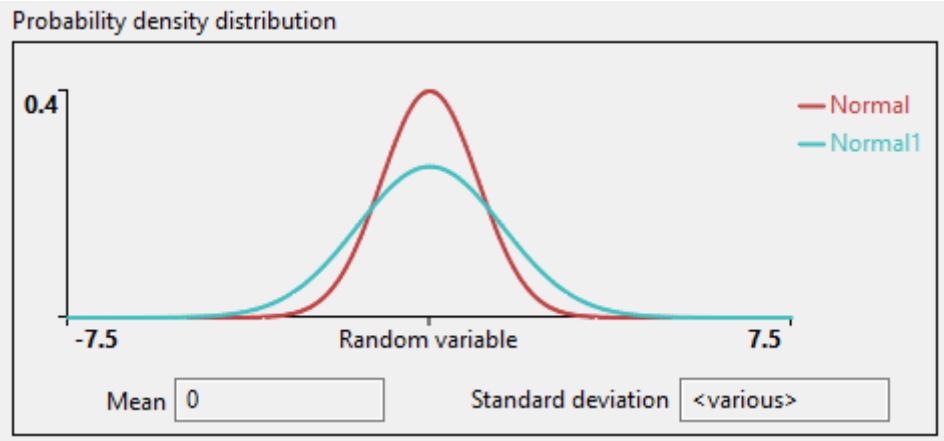
The parameters for the *Piecewise linear* distribution type are specified by the **Segments table**. This table presents the end points of the linear segments of the distribution density curve in two columns. The left column represents the random parameter value. The right column represents the corresponding relative probability density weight. Normalization of the curve is performed automatically.



Rows can be inserted and deleted by commands in the contextual menu of the table. The cell values can be edited by clicking on a cell in a selected row of the table. The *Weight* column accepts any floating-point number > 0. A valid entry in the *Value* column is a floating-point number not smaller than the value on the previous row. Invalid specifications for the latter are taken as equal to the previous row. The table is not available if multiple distribution components are selected.

Probability density preview

The PDF preview chart displays curve plots of the Probability density distribution for distribution components selected in the Component repository. Two text fields at the bottom present the corresponding **Mean** and **Standard deviation** properties.

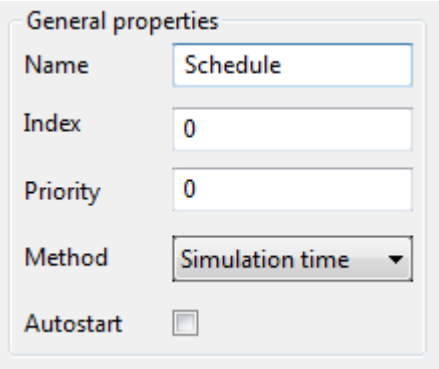


4.2.3. Schedules

Schedule components execute a user-defined Customization script for one or more specified time points in the simulation by a schedule event in the Event calendar.

General schedule properties

The schedule frame is shown if one or more schedule components have been selected in the Component repository. The schedule properties are defined by the **Schedule grid** and a number of additional controls on the frame.



Type name

The Type name property may contain a maximum of 255 characters in the ranges A-Z, a-z, 0-9 and the underscore _ character. The string must start with an alphabetic character and is case-insensitive. The **name** of an object consists of the type name followed by the below-mentioned non-zero index value. The name of each object is unique in the project.

Index

The index property of an object in an integer number >0. When the index property is changed, indexing of the selected objects starts with the specified value, where the index value of non-selected objects with the same type name are skipped.

Priority

Determines the event priority of schedule events created in the Event calendar by the schedule. Possible values: integer number in the range -16000 to +16000.

Method

Selects one of the following schedule methods:

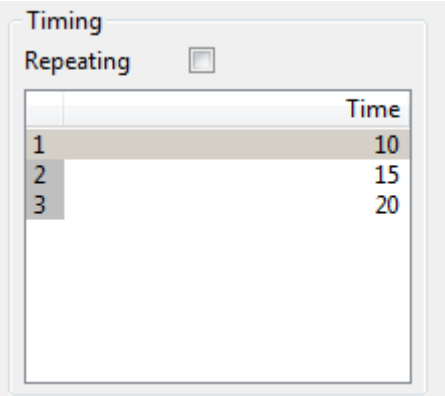
- Simulation time (Default).
  The *Simulation time* method uses the simulation time for schedule timing evaluation. The timing properties of the schedule are specified as simulation time values in the schedule Timing frame if this value is selected for the Method property.
- Simulation clock
  The *Simulation clock* method uses the clock reference frame for the schedule timing control. Selection of this schedule method shows the Recurrence pattern frame in place of the Timing frame. For the *Simulation clock* Method, the schedule is only operational if a dimensional Time unit is selected for the clock reference frame.
- Runtime transition
  Under application of the *Runtime transition* method, execution of the schedule is triggered by transitions in the operation of a simulation, such as the start of a replication. This schedule method does not employ any schedule timing properties, and the Runtime transition frame is shown in place of the Timing frame.

Autostarting

Enables self-initialization of the schedule (Default). Makes the schedule start automatically when a simulation replication starts. A schedule can be activated only by the schedule script method `Run` if this option is deselected.

Schedule Timing

The Timing frame appears when the *Simulation time* value is selected for the schedule Method. The frame displays a list of periodic schedule times. Schedule events are created for each of the simulation times specified in the list.



Schedule times

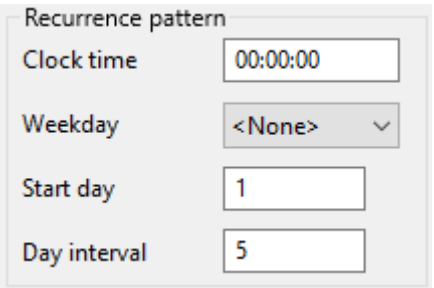
The first column of the schedule times list displays a line number. The second column displays a timing property, which represents the simulation time offset from the start time of the recurrence stream. The start time value is *0* for an autostarting stream. The start time is the current simulation time for a stream started by the schedule `run` command or by an autonomous repetition. Rows can be inserted and deleted by commands in the contextual menu of the list. The timing values be edited by clicking on a selected row in the list. A valid schedule time is a floating-point number >= 0 and not smaller than the value on the previous row. The list cannot be edited if multiple schedule components are selected.

Repeating

Enables cyclic schedule execution. By default, the schedule process is terminated when the last schedule line has been executed. If this property is applied, the schedule is re-run after execution of the last schedule timing.

Recurrence pattern

The Recurrence pattern frame appears when the *Simulation Clock* value is selected for the schedule Method. The frame is enabled if a dimensional Time unit has been specified for the simulation clock. The composition of the frame depends on the Calendar-based clock property. The frame presents the following three recurrence properties if the simulation clock is not Calendar-based.



Clock time

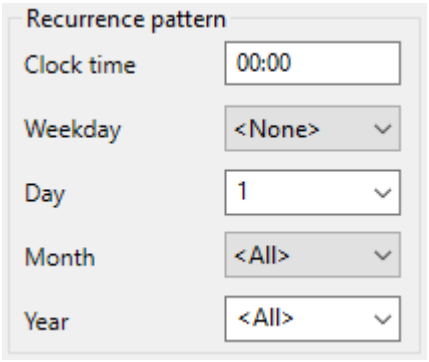
Determines the recurring clock time of the schedule event.

Weekday
Determines the day recurrence pattern of the schedule event. The default value <*None*> signifies that the recurrence pattern is controlled by the below-mentioned Day interval property, ignoring weekdays. The value <*All*> results into daily recurrence.

Start day
Specifies the day number for the initial event in the recurrence sequence as a positive integer number > 0 with default value 1.

Day interval
Specifies the number of days between schedule event recurrences as a positive integer number. The default value 0 results into a single schedule event on the schedule start day. The value 1 results into daily recurrence.

The frame is configured with the following additional recurrence properties for the Calendar-based simulation clock.



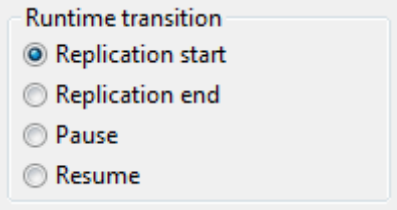
Day
Determines the recurrence pattern for days. The composition of this combo box depends on the selected value of the above-mentioned Weekday property. The control allows selection of a specific month day number in the range 1 - 31 for the weekday value <None>. For other values of the weekday property the control has the possible values <All> (every day), *1st.* to *5th.* (occurrence number in the month) and <*Last*> (last occurrence in the month). The schedule event is skipped if the selected value for the Day property does not occur on the calendar in the month of recurrence.

Month
Selects the month for recurrence of the event. The value <All> results into recurrence of the event for every month in the year of recurrence.

Year
Selects the year for recurrence of the event. The value <All> results into yearly recurrence of the event.

Runtime transition

The Recurrence pattern frame appears if the *Runtime transition* method is selected for the schedule Method. This method will cause the schedule event to be executed when a simulation replication starts, ends, is paused or resumed from the paused state, as indicated in the frame.



Schedule customization

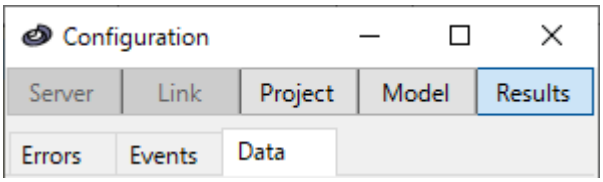
Schedule customization scripts are composed in the Script editor frame.

The **Script editor** frame displays the customization script of the schedule components selected in the Component repository. The text <*various*> appears in the text area if the script code varies between the selected schedules. In that case the contextual menu command **Replace with new script** can be used to create a new uniform script for all selected schedules.

Further instructions for script editing are given in the topic: Script editor. Script coding techniques are discussed in detail in the Customization scripts topic.

5. Results

The configuration window provides access to simulation results and many other elements of the project. The window is opened by the **Configuration** menu on the Tools menu or by double-clicking the viewport.



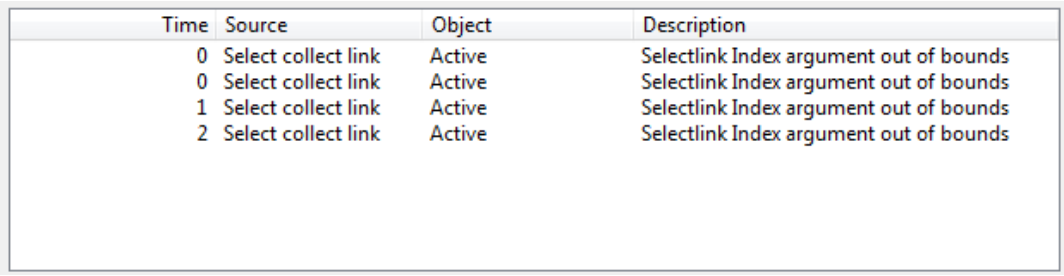
The category bar at the top of window selects the main category for which properties are displayed. Selecting a category displays the corresponding console with all relevant control items.

The **Results** console presents the simulation results organized by the tab strip beneath the category bar:

- **Events** Presents a list of simulation events
- **Errors** Presents a list of simulation runtime script errors
- **Data** Calculated simulation results
-

5.1. Errors

The **Errors** tab on the Results console displays the Error list. The list displays all runtime errors caused by execution of customization scripts in order of occurrence.



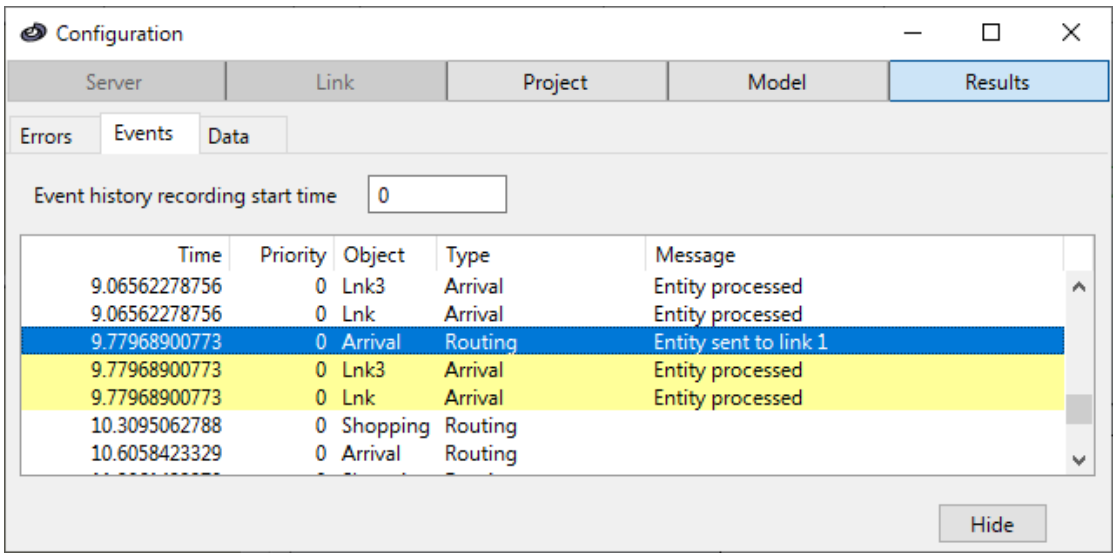
Time	Source	Object	Description
0	Select collect link	Active	Selectlink Index argument out of bounds
0	Select collect link	Active	Selectlink Index argument out of bounds
1	Select collect link	Active	Selectlink Index argument out of bounds
2	Select collect link	Active	Selectlink Index argument out of bounds

The list has four columns:

- **Time** Simulation time of the error occurrence.
- **Source** Name of the object that caused the error.
- **Description** Displays a description of the error number.
- **Object** Name of server or link object of an object customization or the schedule object of a schedule customization that caused the error.
- **Description** Provides information on the nature of the error.

5.2. Events

The **Events** tab on the Results console displays a list of events in the Event calendar, organized in order of creation.



Event history recording start time

Determines the simulation start time of event recording for display on this tab. The text field accepts any floating-point value >= 0. Leaving this field empty (default) results in no events being recorded, such that only future events will be listed.

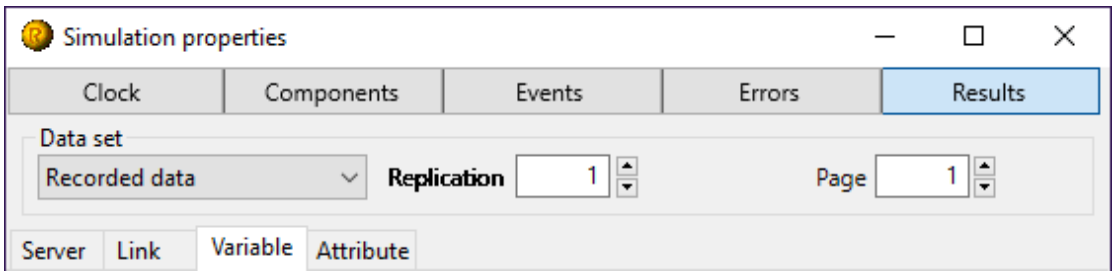
The list has 5 columns each displaying a specific property of the event:

- **Time** Simulation time of the event.
- **Priority** Priority value of the event.
- **Object** Name of the event object. The event object can be a server object, a link object, a schedule component or <system>. The latter signifies the event is not associated with an object.
- **Type** Description of the event type.
- **Message** Provides information on the result of executed events.

Events created by the selected item in the list are highlighted in yellow. The Message column is empty for future events in the calendar.

5.3. Data

The **Data** tab on the Results console provides access to simulation results. The data tab contains the **Data set** frame at the top of the window and the **Object type tab strip** located below the Data set frame.



The results are displayed in data sheets on the Object type tabs. Compilation of these data sheets can take a long time to complete if there are many rows to display. The process can be interrupted and canceled by holding down the **Esc key** while the mouse pointer appears as hourglass. The compilation can be restarted by a click on the sheet.

5.3.1. Data set

The Data set frame contains several items that control which data set is displayed on the tab. These items are visible only if applicable to the data set being presented. The Data set pulldown menu on the left side of the frame selects the type of results to be displayed. The default value *Simulation state* represents current property values in the simulation. The control also offers the selections *Collected statistics* and *Recorded data*. If two or more replications have been completed the selection *Confidence intervals* is available in addition.

Repl.

Specifies a replication number or a range of replications for data review. Only available for the above-mentioned **Data set** selection other than *Simulation state* with simulation in multiple Replications activated. If the replication selector text field is left empty the default value is activated, indicated by a bold font for the label. For the Data set selection *Confidence intervals,* the default value represents all replications. For the other Data sets the default value is the current replication. A range of replications can be selected by entering two replication numbers separated by a dash. In that case the recorded data and collected statistics datasheets have an additional column labeled *Repl*, which displays the replication number of the listed data set.

Page

The page selection control appears if the displayed data set consists of more than 64 columns. The desired section of the data set can be selected by entering the corresponding page number in the text field or by using the adjacent up/down-button.

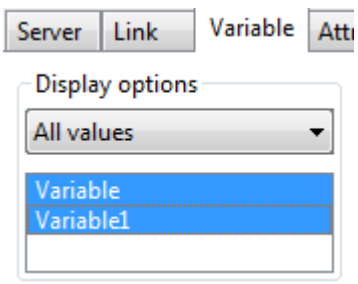
Confidence level

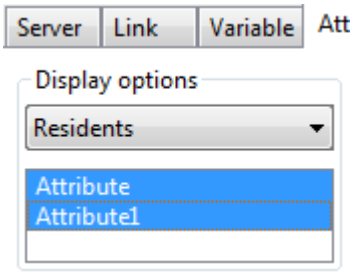
The confidence level to be used in data review by confidence intervals is assigned by entering the desired value in the text field. All values greater than *0* and smaller than *1* are accepted, with default value *0.95*. Only visible if the value *Confidence intervals* has been selected in the **Data set** pulldown menu.

Include pending observation
Includes the current value as additional observation in evaluation of all statistical parameters except Current, for all statistics characteristics other than Residence.

5.3.2. Variable and attribute display options

The Display options frame on the *Variable* and *Attribute* **Object type tabs** offer additional options for component data display. The data display scope is controlled by the selected value of the **pulldown menu** at the top of the frame.





The default item *All values* entails display of both global variable values and local values for server and link objects selected in the viewport. The other selections for variables result into display of either only local or only global values. For attributes, the default value *Residents* presents attribute values of resident entities of server and link objects selected in the viewport. The selection *Entity ID* presents attribute values of all entities that have been assigned an entity Identifier property.

Component data are displayed only for items selected in the **variable/attribute list** of the Display options frame. Components to which the Hide result property is applied do not appear in the list.

5.3.3. Simulation state results

The default value *Simulation state* of the Data set selector presents current object properties.

Server and link simulation state data sheet
The header row of the state data sheet for servers and links indicates the parameters being displayed.

Name	Resident count	Travel count
Lnk	0	0

- ***Name*** Object name.
- ***Resident count*** Number of residents in the server or link.
- ***Travel count*** Number of travels on the link

Variable value data sheet

The header row of the variable value data sheet indicates key values for data stored in variables. The first column displays the variable name, followed by a dot and the server or link object name for local variable values. The cells in other columns represent the actual values stored in the variable components.

Name \ Key		(1)	(2)
Var	1	1	
Var.Srv	11		12

Attribute value data sheet

The header row of the attribute value data sheet indicates the keys for data stored in attributes. The first column displays the attribute name separated by a dot from a qualifier string. For the above-mentioned *Entity ID* display option, the qualifier is the Identifier property of the entity. For the *Residents* display option, the entity qualifier is the container name followed by the hashtag # sign and the entity position in the container object. The cells in other columns represent the actual stored values.

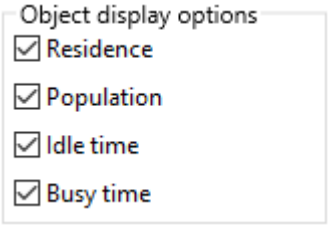
Name \ Key		(1)	
Atr.Srv1 #1	1	2	
Atr.Srv1 #2	1	2	

5.3.4. Collected statistics results

The *Collected statistics* item in the Data set selector presents collected statistics in a data sheet.

Object display options

The display of individual properties is controlled by the **check boxes** in the Display options frame



Server and link statistics data sheet

The **statistics sheet** displays collected statistics data for objects selected in the viewport. The sheet contains a set of 8 statistical parameter values for each of the recorded statistics characteristics. The parameters are indicated in the header row of the sheet. The property column displays the object and the property name separated by a dot.

Property	Current	Observations	Sum	Mean	StdDev	Minimum	Maximum	Rated
Srv.Residence	0.	3	3.	1.	0.	1.	1.	0.3333333
Srv.Population	0	7	3	0.4285714	0.244898	0	1	1.
Srv.Idle		4	0.	0.	0.	0.	0.	0.
Srv.Busy	0.	3	3.	1.	0.	1.	1.	1.

Variable and attribute statistics data sheet

In a similar way as for the above-mentioned object statistics data sheet, the **statistics sheet** displays collected statistics data for components in accordance with the selections made in the display option frame.

Property	Current	Observations	Sum	Mean	StdDev	Minimum	Maximum	Rated
Var(1)	0.8479803	35494	35462.218	0.9991046	0.0988084	-0.1778329	2.3288075	1.000271

The property column displays the component name followed by the key enclosed between parentheses, separated by a dot from the server or link name for local variables, or an entity qualifier for attributes.

5.3.5. Recorded data results

The *Recorded data* option presents recorded data in a data sheet on the Data tab.

Server and link recordings data sheet

The object **recordings data sheet** displays the resident count values as a function of time for objects selected in the viewport.

Time	Srv2	Srv3	
0.9797595	0	0	
0.9797595		1	
1.1884245	1		
2.0804133		2	
2.1928668	2		
2.9104114	3		
3.1280077		3	

The recorded properties are indicated in the header row of the sheet by the object and the property name separated by a dot. Each non-empty cell represents a recording with the simulation time indicated in the first column.

Variable and attribute recordings data sheet

The component **recordings data sheet** displays recorded component values in accordance with the selections made in the display option frame, in a manner similar to the above-mentioned recorded data display for server and link objects. The header row displays the component name followed by the value key enclosed in parentheses, separated by a dot from the server or link name for local variables, or an entity qualifier for attributes.

5.3.6. Confidence intervals

The above-mentioned collected statistics can be examined for variation between replications. Different replications of a simulation that employs distribution components yield different results, because each replication uses a different set of random numbers. In general, the value of a simulated parameter, averaged over all performed replications, approaches the “true” parameter value as the number of replications grows. Because the actual number of replications for a simulation run is finite, there is always some uncertainty in analysis of simulation results with stochastic properties.

The *Confidence intervals* data set option presents an estimation for the confidence interval of the *Current, Mean* and *Rated* statistical parameters, determined on the basis of the Student’s t-distribution. The confidence interval represents a value range which contains, with a specified confidence level, the true parameter value.

The analysis method applied requires the parameter value to be normally distributed in the replications-population. However, there is considerable evidence suggesting that the method is quite robust with respect to deviations from the Normal distribution type in the parameter data.

Confidence interval data sheet

The organization of the confidence interval sheets is similar to the Collected statistics results sheets. The interval is represented in the format *Middle* ± *Range.* The keyword <*Incomplete*> is displayed if there is an inconsistency for a parameter in the collected statistics of the replication series. The column labeled Note Final represents the final value of the replications for the parameter Current.

Property	Final	Mean	Rated
Srv.Residence	0.6774657 ± 0.1698987	0.9986548 ± 0.0034174	0.0009987 ± 0.0000034
Srv.Population	0. ± 0.	0.4997503 ± 0.0000009	1. ± 0.
Srv.Idle	<Incomplete>	0. ± 0.	0. ± 0.

6. Simulation operation

This section covers the operation of simulation models by the following topics:

- Running simulations
- Simulation data
- Simulation display

6.1. Running simulations

This topic discusses the various ways by which a simulation run can be controlled.

Simulation replications

Some statistical data analysis methods in discrete event simulation use multiple equivalent simulations with different random numbers. In Renque such equivalent simulations are known as replications. Replications are individual simulation courses, starting from the *Reset* state, with a unique seed value for the random number generator. As a result, each replication is run with a different set of random numbers, and yields different results. The number of replications to be performed in a simulation run is set by the Replications item of the configuration window. The Confidence intervals option for the simulation results provides a method to inspect simulation data for multiple replications.

Simulation run control

The state of the current replication has one of four possible values: *Reset*, *Running*, *Paused* and *Stopped.* A replication starts from the *Reset* state and has the *Running* state while in operation. A simulation replication terminates and assumes the *Stopped* state when there are no events left in the Event calendar or the predetermined Time limit or End date of the simulation clock have been reached. A simulation run with multiple replications completes all replications successively without interruption, unless paused.

The following items in the Run Menu, of which the first three are also available as simulation toolbar button, are used to control the flow of a simulation run.

- **Reset** Resets the simulation, discarding all data.
- **Restart** Restarts a simulation run, discarding all data.
- **Pause** Pauses or resumes a paused replication. Starts a new replication if the current replication has the *Stopped* state.
- **End replication** Assigns the *Stopped* state to a replication in the *Paused* state.
- **Reset replication** Resets the current replication, discarding all replication data.

The flow of a running simulation can also be controlled by scripting. The script method `Stop` stops a running replication and starts the next replication. The script method `Pause` pauses a simulation. When these methods are used together, the simulation run is interrupted to assume the *Stopped* state.

Mutations made to a project while the current replication is in the *Paused* state are applied when the simulation is resumed, generally without the requirement of a simulation reset.

Time display

The progress of the simulation in time is displayed on the simulation toolbar, by default in a format that best fits the properties selected for the simulation clock. Time display is adaptable by text markup in the time display format string, which is edited by clicking on the Simulation time panel of the simulation control toolbar.

The simulation time can also be displayed in Symbol text, as specified by the String text edit field in the *Symbol text* scope of the Text tab of the configuration window.

Any combination of escape sequences can be used in both format strings. All characters that do not represent an escape sequence are displayed as literal text.

6.2. Simulation data

Renque automatically keeps track of the simulation time at which an entity was stored as resident in a server or link object, and of the number of residents and travels. Additional data monitoring options are available through statistics collection and recording of certain property values in time. Simulation data are accessed on the on the Data tab of the configuration window.

6.2.1. Data storing

Variable and attribute components are used to store simulation data. Data values are assigned and retrieved by scripting. Both components can store the scripting language data types integer, floating point, string and boolean, as well as Renque object instances except for variables and attributes. Storage of other data types, such as arrays, instances of user-defined script classes, Python tuples or complex numbers are not supported. If an entity or another Renque object stored in a variable and attribute component is destructed in the model it is also removed from the component.

Simulation data and most Renque object types can also be stored in script language variables, that are declared within the scope of a customization script. However, this practice is not recommended for Renque entity objects, because reference to such objects can be lost when stored in another location than in a Renque object (Server, Link, Variable or Attribute). When this happens, subsequent attempts to access any property of the entity object in the script will raise a runtime error with a description containing the phrase: *'Invalidated entity reference'*.

6.2.2. Statistics collection

Renque can collect statistics for viewport objects and variable and attribute components. Statistics collection is performed for the following characteristics:

- **Residence** Entity storage time in the host object.
- **Population** Number of entities stored in the object.
- **Busy** Duration of the *Busy* status.
- **Idle** Duration of the *Idle* status.
- **Valuation** Numeric value stored in a variable or entity attribute.

The **Population** and **Valuation** qualities are logged when the value is about to change. The **Busy** and **Idle** properties are logged when the corresponding object status ends. The **Residence** time is logged when a resident entity is removed from storage in a server or a link. The Residence characteristic is not available for block and fusion servers. If the Include pending observation option is applied, the reported statistics include the value of the Current parameter as additional observation, for all characteristics except Residence**.** Subsequent assignments of the same value to a variable and attribute are logged as individual Valuation observations. The default value is not logged as initial observation.

Statistics collection is optional and deactivated by default. Attribute statistics for multiple replications are collected only for entities that have been assigned an Identifier property, as this property is used to inspect attribute data for previous replications for which the simulation data have been discarded. Collection is activated for individual objects items on the user interface:

- The Collect statistics item for servers and links of the configuration window.
- The Collect statistics item for components of the configuration window.

Statistics collection starts immediately at the start of a simulation replication and continues until the replication ends. The `StatisticsReset` script method discards all observations and recommences the statistics collection process.

Statistical parameters

A logged property value is referred to as an observation. The following statistical parameters are calculated for the collected observations:

- **Current** Present property value awaiting to be logged as observation.
- **Count** Number of observations: N.
- **Sum** Sum of the observations Σ X.
- **Mean** Algebraic mean of the observations: $\mu = \Sigma X / N$.
- **StdDev** Standard deviation of the observations: $\sqrt{(\Sigma (X - \mu)^2 / N)}$
- **Minimum** Minimum value of the observations
- **Maximum** Maximum value of the observations.
- **Rated** Property-dependent fractional parameter.

The significance of the *Rated* property varies between the recorded properties:

- **Holdup** Ratio of the Mean ***Residence*** parameter to the total simulation time of the replication.
- **Time-averaged** Time-averaged value: $\Sigma (X \cdot dt) / \Sigma dt$, where dt is duration of observation X for the ***Population*** property and the ***Valuation*** property of variable and attribute components.
- **Fraction** Ratio of the status time sum ΣX_A of the ***Busy*** or ***Idle*** property *A* to the status time sum of the Busy and Idle properties combined $\Sigma X_A + \Sigma X_B$.

The parameter Current signifies the pending observation, i.e. the value to be logged if observation logging were imminent. The value of the Current parameter of the **Population** property is identical to the value returned by the `ResidentCount` function. The Current parameter of the **Residence** property presents the mean residence time of all entities stored in the server or link. The current Residence is *Nil* if the object has no residents. The current value of the **Idle** and **Busy** properties is *Nil* if the property doesn't represent the current object status. The current value of variable and attribute component **Valuation** property is *Nil* if the stored value is not numeric.

Collected statistics can be examined on the Results console of the configuration window by selecting the *Collected statistics* item from the Data set dropdown menu on the Data tab. Statistics data are also accessible by scripting through statistics methods.

6.2.3. Data recording

Data recording is used to trace the history of certain properties in a simulation run. Data recording can be activated for the Population property of viewport objects and the Valuation property of Attribute and Variable components.

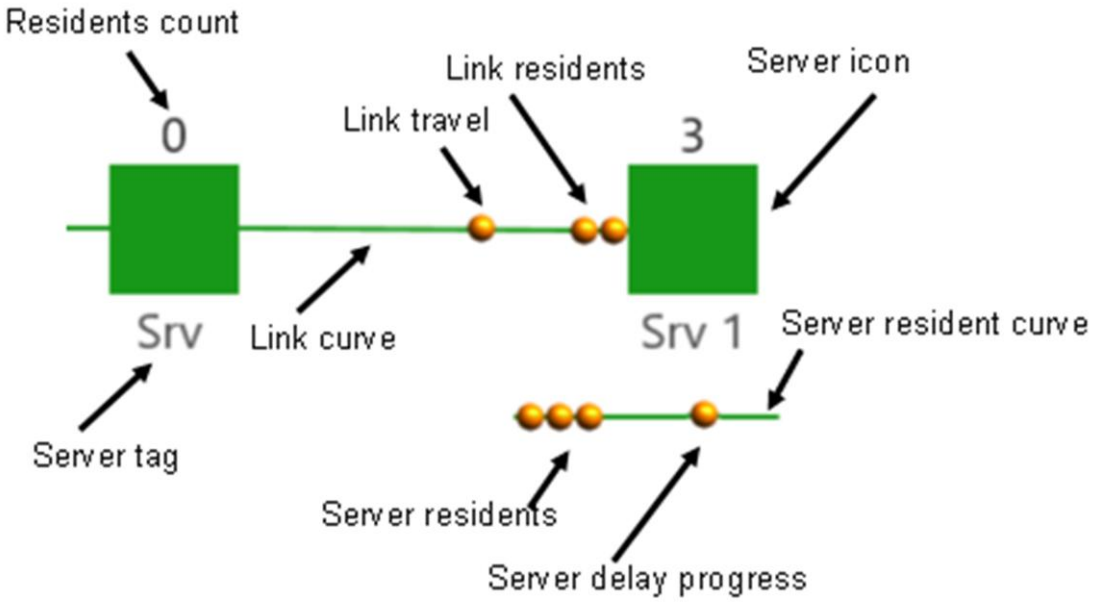
Data recording can be considerably more memory consuming than the above-mentioned statistics collection method. Data recording is activated for individual objects on the user interface as follows:

- The Record data item for viewport objects of the configuration window.
- The Record data item for components of the configuration window.

Recorded data results can be examined on the Results console of the configuration window by selecting the *Recorded data* item from the Data set dropdown menu on the Data tab.

6.3. Simulation display

A Renque simulation can be visualized through a number of display elements in the viewport. Simulation visualization can be employed as visual model design tool and to support presentation. The figure below demonstrates which server and link elements can be displayed or hidden during a simulation run.



The simulation display properties of individual server and link objects are controlled in the Server simulation display options and the Link simulation display options frames of the configuration window.

Animation characteristics

Renque can animate entity link travel and the progress of the delay period of server residents. The animation speed is controlled by the Animation speed slider item on the Menu bar. Animation can be toggled on/off by the Animation button on the simulation toolbar.

Entity travel on links is animated with a uniform apparent travel time, as determined by the Link travel animation timing property. The simulation time does not advance during travel animation. The entities of multiple travels are animated as a lined-up series of entity icons on the link curve. The maximum number of travels displayed on a link is limited by the length of the curve.

Server resident retention progress is animated by motion of entity icons along the residents-curve of the server. The relative position of the entity along the curve reflects the completed residence time with respect to the Delay property applied to the entity. This animation feature is activated by the Animate retention server option.

Resident entities

Entities stored in a server or in a link are painted as a queue on the server residents-curve or link curve. The number of entities displayed is limited by the length of the curve. No residents are shown if animation is disabled.

Caption

Text can be displayed on the view port by several constituents. Server and link objects both have a Tag property, which usually indicates the object name. The tag is not displayed for block servers. Servers can also display text in the server icon. The appearance of either of these caption items is controlled on the Text tab of the configuration window. The server method `Text` can be used to modify the symbol text at runtime. The number of server residents is displayed above the server image. This counter has a fixed style and format and is not shown if the simulation is in the reset state or if animation is disabled.

Server image

The appearance of the server image is controlled on the Display tab of the configuration window. The server method `Icon` can be used to modify the server image at runtime.

Display order

Server resident curves are displayed on top of the server image. Link curves are displayed on top of all servers. The z-order level of servers can be modified by the **order** menu of the Format Menu. The display order of link curves cannot be changed.

7. Customization scripts

Customization of a simulation for link and server objects and for schedule components is performed by scripts. A customization script is a small program written in scripting language.

Both server and link objects are assigned a default customization script upon creation of the object. Customization scripts can be modified in the configuration window on the Custom tab for servers and links and on the Components tab for schedules.

The scripting language to be used in the project is selected by the Scripting language item in the preferences utility. Renque has a choice of two script languages:

XojoScript This script language is a dialect of the traditional programming language BASIC, which is a widely used high-level programming language, known for it's ease of use.

Python Python is an interpreted, object-oriented, high-level open-source programming language with dynamic semantics, which after years of steady growth has become one of the most used programming languages today.

Standard VB The original condensed version of the programming language BASIC, compatible with Microsoft's *VBscript* language.

One does not require extensive knowledge of the scripting language nor does one need to be familiar with programming techniques to use customizations in a Renque project. The following examples illustrate the type of Basic scripts typically used in a Renque model.

```
Srv.ArrivalAccept
```
invokes the `ArrivalAccept` method of a server named *Srv*.

```
Var = Var + 1
```
increments the global value of the variable named *Var*

```
if Var = 2 then Srv.Text = "Failed"
```
makes the server named *Srv* display the text '*Failed*' only if the value of variable *Var* equals *2*.

This section focuses on the use of scripts in the Renque simulation environment. General information on the *XojoScript* language can be obtained through the web link: https://docs.xojo.com/UserGuide:XojoScript_Language.
For a guide on using Python the reader is directed to:
https://docs.python.org/3/tutorial/index.html.

The Renque scripting syntax topic provides general information about scripting within the Renque environment. Further details on the unique set of Renque script methods, which has been carefully devised to offer maximum control of the simulation with just a small number of commands are discussed by context in the following topics:

- General methods
- Entity methods
- Server methods
- Link methods
- Component methods
- Statistics methods
- Object referencing functions

The notation used in these topics specifies the method name, followed by the method arguments enclosed in parenthesis. The method type (Procedure, Function or Property) is given, together with the return value type for functions and properties. A complete list of the available Renque methods in alphabetical order is given in the Reference section.

7.1. Renque scripting syntax

7.1.1. Object references

Scripts can be used to assign and read object properties and to impose certain transformations on objects in the simulation project.

Direct object references

All server, links and components can be accessed in scripts by their Name property. A direct reference by name is preserved in the script if the name property of the object is changed in the sense that the name also changes in the script. Using shared class methods objects can also be accessed by the Type name and Index properties or by iteration.

Indirect object references

Some Renque methods return a reference to a viewport object, a component or an entity. The server script function `Links()`, for example, returns a link object connected to the server. Links have similar methods that return a connected server object. Entity objects can only be referenced indirectly. The shared server and link function `Residents()`, for example, returns a resident entity of the object.

Variable and attribute references - key values

Variable or Attribute objects, which are used to store data, are referenced by name. A directly referenced variable or attribute object such as in the code line:

```
Vrb1 = Vrb2
```

automatically references the value stored in the variable objects `Vrb1` and `Vrb2` as opposed to the object itself. The statement results into assignment of the **value** of one variable object to the **value** of the other.

A variable or attribute object returns the default value of the component until a different value has been assigned. Values can be stored in a variable or an attribute with optionally an integer or string value, or a series of integer values as key. The code statement:

```
Var(1,2,2) = "A"          (BASIC)
Var[1,2,2] = "A"          (Python)
```

stores the string value "A" in the variable named *Var* with key combination *(1,2,2)*. An unlimited number of integer keys can be passed, separated by commas. Note that square bracket pairs [ ] are used instead of parentheses for specifying keys in the Python scripting language. Note also that it is not necessary to dimension variable and attribute objects for keys, as opposed to array variables defined in a script, such as for example in the Basic expression:

```
Dim X(3) as Integer
```

which declares a variable within the scope of the customization script.

Local variables are accessed by a server or link object reference followed by the variable name, separated by a dot: For example:

```
Srv.Vrb("Keystr") = 3          (BASIC)
Srv.Vrb["Keystr"] = 3          (Python)
```

assigns the value *3* to the variable named *Vrb* of the server named *Srv* with key value *"Keystr".*

Attribute values can only be accessed by an entity reference followed by the attribute name, separated by a dot: For example:

```
Object1.Residents(1).Atr = 3
```

assigns the value *3* to the Attribute named *Atr* of the first resident of the server or link object named *Object1*, without any key value.

Type casting in BASIC

The Basic language requires a method called casting to access elements of an object of which the type is undetermined, such as when stored in a Variable or Attribute component. Casting involves a class identifier that explicitly expresses the object type. For example:

```
Entity(Var).EntryTime
```

returns the EntryTime property of an entity that is stored in variable component *Var*. Omission of the `Entity()` statement in the previous expression would cause a compilation error because the return value type of a variable component is not an entity but a variant. Execution of the above expression causes a runtime error if Var does not contain an entity. The IsA operator can be used to verify an object type.

Implicit signal object references

Renque signal properties are accessible through the Signal read-only property of the Sim object, using the following expression:

```
Sim.Signal
```

The **signal recipient** object is retrieved by the expression:

```
Sim.Signal.Recipient()
```

The parentheses pair following the Recipient function may be omitted within the BASIC script language. Methods of the signal recipient may be referenced implicitly by omission of the recipient reference. For example:

`Sim.Signal.Recipient().Icon` as well as `Srv.Icon`

are both equivalent to

```
.Icon
```

in the customization script of the server named *Srv*. The dot symbol of the implicit reference must be preceded by a space or a symbolic operator, such as a plus sign a parenthesis symbol, or a comma.

The **signal entity** object is returned by the general simulation method `Entity` as in the expression:

```
Sim.Signal.Entity()
```

This entity object may be referenced implicitly by omission of the Entity reference altogether. For example, the statements:

`Sim.Signal.Entity().Icon = BlueBall` vs `Icon = BlueBall`

are equivalent. The implicit entity referencing syntax also works for Attributes. The parentheses pair following the `Entity` function may be omitted within the BASIC script language. Note that the signal entity is *Nil/None* for some signal types.

7.1.2. Renque scripting methods

Renque offers an assortment of script methods that provide access to the properties of a simulation project and its elements.

In accordance with the syntax rules of the Basic and Python languages, the general form of a Renque script method is given by:

```
MethodName[(argument1,[argument2,[...]])]
```

A method may or may not require one or more arguments, separated by commas. The square brackets in this definition indicate optional arguments. A default value is used if an optional argument is omitted in a method call. The default value is *0* for the integer and floating-point arguments, *False* for a boolean and *Nil* for an object.

Scripting in Renque has three method types.
1. The *Procedure* method type, which does not have a return value
2. The *Function* method type, which does have a return value
3. The *Property* method type, which can be used to both assign and evaluate property values.

The syntax structure of three method types is as follows:

```
ProcedureName[(…)]

… = FunctionName[(…)]

PropertyName = …    and    … = PropertyName[(…)]
```

In the Basic language parentheses are optional for procedure calls and for function and property calls without arguments. In the Python language, the use of parentheses in calling Renque functions and procedures is required even if the method call has no arguments. In contrast, parentheses cannot be used in addressing the property method type in Python. In addition, any argument defined for a Renque object property, such as the InitValue argument for the server Icon property, cannot be used in Python.

Most Renque object classes have a number of methods specific to the class. These methods are referred to as Renque object methods. The `Origin` method, for instance, is a method only defined for links. Renque also has a variety of functions and procedures which address general aspects of the simulation without any particular object type context.

7.1.3. The Renque root module: Sim

the instruction `Sim` references the Renque module Sim. This module is the root for access to all general simulation methods, which are not in the scope of a particular object type, such as the server or entity. For example, the simulation method `Pause` is executed by the syntax:

```
Sim.Pause()
```

The Sim module also contains the following three read-only properties, which convey further arrangement of general simulation methods:

- `Clock`
- `Signal`
- `Animation`

Each of these three properties are singleton class instances, which cluster a set of interrelated simulation methods. The `Clock` property, for example, provides access to all available methods associated with the simulation clock, such as in the expression:

```
Sim.Clock.Replication()
```

which returns the current replication number.

The distribution of simulation methods in this manner, using dot notation, makes it easy to find Renque commands among the many keywords of the scripting language itself and improves the readability of customization scripts

7.1.4. Runtime errors

A script that contains a syntax error is said to be invalid. Invalid scripts are not executed in a simulation run. Valid scripts can still cause errors in a simulation run for a variety of reasons. Such errors are known as Runtime errors. Runtime errors do not cause a simulation run to be stopped. If a runtime error occurs, the execution of the script causing the error is aborted, but the simulation will continue as normal.

Error messages

All runtime errors create an entry in the Errors list, for review by the project designer. The list may contain general script language error messages and Renque-specific error messages.

7.1.5. Notes

When using the Python scripting language in Renque the following particularities must taken into account:

- The use of Python with Renque requires a correctly installed Python with version number 3.7 or higher. A free Python installer can be obtained at https://www.python.org/downloads/.
- The values passed to arguments of Renque methods must be of the correct type, even though Python is dynamically typed.
- Variable and Attribute keys are specified with square brackets [ ] as opposed to parentheses for Basic.
- Parenthesis are required for Renque function and procedure calls even if no arguments are passed.
- Renque object property calls cannot have parentheses or arguments in Python. The distribution property Parameters assigns and retrieves all parameter values of the component as a Python tuple.

Some notes for the reader who is familiar with other BASIC script dialects than XojoScript:

- The For...Next loop construction in the XojoScript language re-evaluates the final value of the counter in every loop iteration.
- Late-binding of variables is not supported.
- Automatic type casting from the floating-point to the integer variable type results into round-off to the nearest whole number towards zero.

7.2. General Renque methods

General Renque methods, which are not a method of a server, link or component are accessed through simulation keywords.

`Sim`
Read only property (object). Root of all general Renque methods.

7.2.1. Read-only properties

The Sim keyword carries the following read-only properties, which provide access to specific subsets of related simulation methods:

`Animation`
Read-only property (object). Required to call animation methods using the expression `Sim.Animation.`

`Clock`
Read-only property (object). Required to call clock functions using the expression `Sim.Clock.`

`Signal`
Read-only property (object). Required to call signal methods using the expression `Sim.Signal.`

7.2.2. General simulation methods

General simulation methods are addressed as method of the `Sim` object keyword.

`Pause`
Procedure. The `Sim.Pause()` statement pauses the simulation after execution of the current calendar event.

`Stop`
Procedure. `Sim.Pause()` statement forces termination of the present replication after execution of the current calendar event.

`Time`
Function (Floating-point). The expression `Sim.Time()` returns the simulation time as a floating-point number.

`Random`
Function (Floating-point). The expression `Sim.Random()` returns a random number between 0 and 1.

`StatisticsReset( [Collector as Object])`
Procedure. The `Sim.StatisticsReset()` statement clears all collected statistics data for the present simulation replication and reinitializes data collection. If a server, link, variable or attribute object is passed to the optional **Collector** argument, the reset action is applied only to the object. The statistics collection start time is re-assigned if the Collector argument is omitted.

7.2.3. Signal methods

The signal properties of the active signal are accessed using the `Sim.Signal` syntax.

`Recipient`
Function (Object). Returns the server or link object of an object customization or the schedule object of a schedule customization being executed for the active signal. The signal recipient can also be implicitly referenced by a dot.

`LinkPosition`
Function (Integer). Returns the connection position of the link involved with the signal. The function returns the 1-based position in top-down connection order of the link. Negative values represent upstream links of the signal server and positive values represent downstream links. The function returns the value *0* if the signal incident has no specific link involvement.

`Entity`
Function (Entity). Returns the entity object involved with the signal. The function returns the value *Nil* if the signal does not involve an entity. All properties of the signal entity can be implicitly referenced as global method.

`IsArrival`
Function (Boolean). Returns *True* if the signal type is a server Arrival signal.

`Priority`
Function (Integer). Returns the priority value of the current event, which has triggered the signal.

7.2.4. Animation methods

The animation properties of the simulation are accessed using the `Sim.Animation` syntax.

`Freeze( Timing as Double = 1, [Realtime as Boolean])`
Procedure. Updates the viewport and freezes the simulation for a time period determined by the argument **Timing**. If the value *True* is passed for the optional argument **Realtime** the Timing argument represents the time period in milliseconds real time. Otherwise the Timing argument represents a proportionality factor to the applied Travel animation time. Not effective if animation has been deactivated or the below-mentioned suspended animation condition has been activated.

`Suspended`
Property (Boolean). Returns or assigns the suspended animation condition. If the value *True* is assigned animation is suspended until the value *False* is assigned or until the next replication starts. Only effective if animation is activated.

`LastUpdateTime( [Realtime as Boolean])`
Function (Floating-point). Returns the time passed at completion of the last viewport update since the start of the current simulation replication or resumption from the paused state. The difference in return value for two calls of this function can be used to determine how long a certain display feature has been visible on the screen and subsequently decide if the above-mentioned `Freeze` command should be used to extend this period. The optional argument **Realtime** has the same purpose as for the `Freeze` procedure. The function returns the value *0* only if animation has been disabled.

7.2.5. Clock functions

Properties of the simulation clock are accessed using the `Sim.Clock` syntax.

General clock functions

`SimTimeLimit`
Function (Floating-point | Nil). Returns the Time limit for simulation replications as floating-point number. The function returns *Nil* if no Time limit is specified.

`Replication`
Function (Integer). Returns the current replication number

`ReplicationCount`
Function (Integer). Returns the total number of replication runs to be performed for the current simulation.

Clock time functions

Clock time functions return information of the simulation clock in relation to the clock time. An error is raised when a Clock time function is called and the Time unit of the clock is *Dimensionless*. The available Clock time functions are:

`StartWeekday`
Function (Integer). Returns the day number of the clock start day: 1=Sunday ... 7=Saturday.

`Time`
Function (String). Returns the clock time in the short time-format of the system.

`TimeLong`
Function (String). Returns the clock time in the long time-format of the system.

`TimeWeekday`
Function (Integer). Returns the current day number of the clock: 1=Sunday ... 7=Saturday.

`TimeHour`
Function (Integer). Returns the hours number of the current clock time (0-23).

`TimeMinute`
Function (Integer). Returns the minutes number of the current clock time (0-59).

`TimeSecond`
Function (Integer). Returns the seconds number of the current clock time (0-59).

`Weeks`
Function (Integer). Returns the week count in the calendar.

`Days`
Function (Integer). Returns the day count in the calendar.

`Hours`
Function (Integer). Returns the hour count in the calendar.

`Minutes`
Function (Integer). Returns the minute count in the calendar.

`Seconds`
Function (Integer). Returns the second count in the calendar.

Clock Calendar functions

Calendar functions return information of the simulation clock in relation to the calendar. An error is raised when one of the calendar functions is called and the simulation clock is not Calendar-based. The available clock calendar functions are:

StartDate
Function (String). Returns the simulation Start date in the date format of the operating system.

EndDate
Function (String). Returns the simulation End date in the date format of the operating system. The function returns an empty string if no end date is specified.

Date
Function (String). Returns the clock date in the short date format of the system.

DateLong
Function (String). Returns the clock date in the long date format of the system.

DateYear
Function (Integer). Returns the year number of the current clock date.

DateMonth
Function (Integer). Returns the month number of the current clock date.

DateWeek
Function (Integer). Returns the week number in the year of the current clock date.

DateDay
Function (Integer). Returns the day number of the month of the current clock date.

DateDayOfYear
Function (Integer). Returns the number of days into the year of the current clock date.

7.3. Entity methods

Entity methods are used to obtain information on the status of an entity or to cause the entity to undergo some transformation. The following Entity methods are defined:

Operational commands

`Transfer([RecipientLink as Link], [Priority as Integer])`
Procedure. Moves the entity to the link object specified by the **RecipientLink** argument for travel. The optional argument **Priority** determines the priority of the arrival event created for the entity. If omitted, the priority of the current event is used. The entity is destructed if the RecipientLink argument is *Nil*. The entity is removed from its container if stored in server or a link, and travel is aborted if the entity is travelling on a link. The Entity functions `ResidentPosition` and `Host` can be used to determine the location of an entity.

`Route( [DelayTime as Variant], [LinkPointer as integer], [SignaledDeparture as Boolean])`
Procedure. Routes an entity at a server. The method invalidates a previously created routing event for the entity if the entity is a server resident. If the entity is a link resident or travel entity, it is stored as resident in the destination server of the link. A runtime error is raised if the entity is travelling on a downstream open link. The optional argument **DelayTime** determines the applied Delay property value as a floating-point number >= *0*, ignoring any residence time already spent by the entity in its' host server. No departure event is created for the default value *Nil* of the DelayTime argument. The optional argument **LinkPointer** determines the 1-based connection position of the downstream recipient link for the entity. The default value *0* selects the target link as determined by the Routing link property of the server. The value *-1* results into destruction of the entity upon departure. Passing a value for the optional argument **SignaledDeparture** value overrides the Signal departures server property for the routed entity.

Miscellaneous methods

`Icon`
Property (Picture). Returns or assigns the Icon property of the entity as picture object. The value *Nil* represents the default entity icon.

`Identifier`
Property (String). Returns or assigns an identifying string for an entity. The assigned value string can be any non-empty string. The assignment raises an error if the value is the Identifier of another existing entity, or if the entity already has an Identifier value.

`EntryTime`
Function (Floating-point). Returns the simulation time of the last occasion for which the entity was created or stored as a resident in a server or link.

`DepartureTime`
Function (Floating-point). Returns the simulation time of the event associated with release of the entity from a server. The property returns *Nil* if the entity is not stored in a server or does not have a departure event.

`Host`
Function (Server|Link). Returns the server storage or the link storage or travel host of the entity. The below-mentioned function `ResidentPosition` can be used to determine the location of the entity when stored as resident in a server or link host.

`ResidentPosition`
Function (Integer). Returns the sequence position of the entity as `Resident` in a server or link. The function returns the value *0* if the entity is not stored in a server or a link.

7.4. Server methods

Server methods are called on a server object. They are used to obtain information on the status of the referenced server or to cause the server to undergo some transformation. The following server methods are defined:

Operational commands

`ArrivalAccept`
Procedure|Function (Boolean). Processes an arriving entity for routing in accordance with the applied server service properties *Delay* and *Routing link* and *Signal departures*. Returns the value *True* is an arriving entity was handled successfully. The method is ineffective and returns the value *False* if called in a non-Arrival signal. The method raises a runtime error if called on another object than the signal recipient, or if the arriving entity has already been processed by any prior command in the customization script. The routing properties assigned by the method can be modified at any time for a server resident, using the entity `Route` command.

`EntityProcure( [SkippedLinkCount as integer], [ResidentPreferred as Boolean])`
Procedure|Function (Entity). Seeks an upstream connected link in top-down connection order to provide an entity for travel. The entity to be provided is either the first resident of a link's origin server, or a new entity created on an upstream open link. All link resident entities are ignored. The optional argument **SkippedLinkCount** determines the number of items skipped from the top in the link search (if positive) or down to the bottom (if negative). If the value *True* is passed for the optional argument **ResidentPreferred** the method only creates an entity on an upstream open link if no residents are encountered on the origin server of all other examined links. The function returns the supplied travel entity, if successful. The method does not trigger a departure signal for entities procured from servers. In BASIC, the method may also be employed as a procedure (without `Call` keyword), as long as no arguments are passed.

`SignalSend( [LinkPosition as integer])`
Procedure. Creates an event that signals the server for execution of the object customization. The optional argument **LinkPosition** assigns the LinkPosition property of the signal in accordance with the Position argument of the `Links` function. The default value *0* expresses that the signal link is undetermined. For other values, a runtime error is raised if the argument does not match a link connection.

`ResidentSwap (Position1 as Integer, Position2 as Integer)`
Procedure. Switches the positions of two resident entities stored in the server, as specified by the arguments **Position1** and **Position2**. The arguments represent the storage position as for the Position argument of the below-mentioned `Residents` function. A runtime error is raised if the value passed for an argument does not match a resident.

Property assigning methods

`Icon( [InitValue as Boolean] )`
Property (Picture). Returns or sets the Icon property of the symbol server as picture object. The property returns the initial value of the reset state if the value *True* is passed for the optional argument **InitValue**. The InitValue option cannot be used for property assignment or in Python. The value *Nil* corresponds to no picture displayed on the server symbol.

Text([InitValue as Boolean])
Property (Variant). Returns or assigns the Symbol text property of a server. The property returns the initial string value of the reset state if the value *True* is passed for the optional argument **InitValue**. The InitValue option cannot be used for property assignment or in Python. The property assignment causes a runtime error if the assigned value is not a string and cannot be converted to a string by the script engine.

Priority([InitValue as Boolean])
Property (Integer). Returns or assigns the Priority server property. The property returns the initial property value of the reset state if the value *True* is passed for the optional argument **InitValue**. The InitValue argument cannot be used for assignment of the property or in Python.

Miscellaneous methods

Index
Function (Integer). Returns the index property.

Name
Function (String). Returns the name of the object, which is the Type name concatenated with the Index value.

Delay
Function (Variant). Returns the value of the Delay server property. The function returns the evaluation of a component assigned to the Delay property, not the component itself. Evaluation of an attribute component assigned to the Delay property is performed for the signal entity. In that case, if the signal entity is *Nil,* the default value of the attribute is returned while creating a warning in the runtime error list. If evaluation of an assigned component yields a non-numerical value or a value *< 0* the function returns *Nil*.

Links(Position as Integer)
Function (Object). Returns a link object connected to the server. The argument **Position** denotes the 1-based link position in top-down order of connection. A negative Position value references upstream links of the server and a positive value references downstream links. A runtime error is raised if the value passed for the Position argument does not match a link connection. The link methods `OriginPosition` and `DestinationPosition` can be used to obtain the connection position of a link.

LinkCount([Upstream as Boolean])
Function (Integer). Returns the number of downstream connected links to the server. Returns the number of upstream connected links to the server if the value *True* is passed for the optional argument **Upstream**.

Residents(Position as Integer)
Function (Entity). Returns a resident entity of the server. The resident entities are positioned in the server in order of arrival. The argument **Position** denotes the 1-based entity position. A runtime error is raised if the value passed for the Position argument does not match a resident entity. The entity method `ResidentPosition` can be used to determine the position of a specific entity in a server.

ResidentCount
Function (Integer). Returns the number of entities stored as resident in the server.

Chart
Read-only property (Object). Returns the chart object of a symbol server for graphing of simulation data.

Statistics
Read-only property (Object). Required to obtain collected statistics through the statistics characteristics of the server.

Chart methods

DataAdd(X as double|String, Y as double, [SeriesIndex as Integer = 1])
Procedure. Adds data to the chart object of a symbol server. The procedure appends a plot value pair (X, Y) to the chart if a numeric value is passed for the argument **X**. Passing a string value for **X** assigns the value of argument **Y** in a column chart to the category specified by argument X. If the category addressed by the X-argument does not exist, it is created. The special value *Timestep* (case-insensitive) results into addition of the value Y to a line chart of step changes in Y as a function of the simulation time. The optional argument **SeriesIndex** represents the 1-based sequence number of the chart series to which the data is added. A temporary new series is created if none exists for the value of SeriesIndex, to be removed again when a simulation replication is reset.

DataClear([NewMinX as double])
Procedure. Removes data from the chart object of a symbol server. Only the values of a chart are removed that have an x-coordinate smaller than the value passed for argument **NewMinX**. If the argument is omitted, all data are removed from the chart, including category data.

DataClear([Category as String], [SeriesIndex as integer])
Procedure. Removes category data from the of a symbol server. The optional argument **Category** specifies the category for which the data is to be removed. An empty string removes all category data. The optional argument **SeriesIndex** represents the 1-based sequence number of the chart series from which the data is to be removed. The default value 0 for the argument results into all series of the chart being subjected to the command.

Refresh
Procedure. Forces a display update of chart object of a symbol server in the viewport, to which the *Block auto-refresh* option is applied.

7.5. Link methods

Link methods are called on a link object. They are used to obtain information on the status of the link object or to cause the link to undergo some transformation. The following link methods are defined:

Operational methods

`EntityCreate( [Source as Entity], [Priority as Integer])`
Procedure|Function (Entity). Creates an entity on the link for travel and returns the entity object. The function returns a copy of the entity object passed to the optional argument **Source**. The copy procedure includes the Icon property and all attribute values. A new entity is created if the Source argument is omitted. In both cases the `EntryTime` property of the returned entity is assigned the current simulation time. The optional argument **Priority** determines the priority of the arrival event created for the entity. If omitted, the priority of the current event is used. In BASIC, the method may also be employed as a procedure (without Call keyword), provided no argument value is passed.

`ResidentSwap( Position1 as integer, Position2 as integer)`
Procedure. Switches the positions of two resident entities of the link, as specified by the arguments **Position1** and **Position2**. The arguments are positive numbers representing the storage position as for the *Position* argument of the `Residents` function. A runtime error is raised if the value passed for an argument does not match a resident.

Connectivity methods

`Origin`
Function (Server). Returns the origin server object of the link. The function returns *Nil* for upstream open links.

`Destination`
Function (Server). Returns the destination server object of the link. The function returns *Nil* for downstream open links.

`OriginPosition`
Function (Integer). Returns the 1-based link connection position at the Origin server. The function returns the value *0* if the server has no Origin server.

`DestinationPosition`
Function (Integer). Returns the 1-based link connection position at the Destination server. The function returns the value *0* if the server has no Destination server. The function returns a negative value for consistency with the Position argument passed to the Links function to address upstream links of a server.

Miscellaneous methods

`Index`
Function (Integer). Returns the index property.

`Name`
Function (String). Returns the name of the object, which is the Type name concatenated with the Index value.

`Residents(Position as integer)`
Function (Entity). Returns a resident entity of the link. The resident entities are positioned in the object in order of arrival. The argument **Position** denotes the 1-based entity position. A runtime error is raised if the value passed for the Position

argument does not match a resident entity. The entity method `ResidentPosition` can be used to determine the position of a specific entity in a link.

`ResidentCount`
Function (Integer). Returns the number of entities stored as resident in the link.

`TravelCount`
Function (Integer). Returns the number of entities travelling on the link.

`Statistics`
Read-only property (Object). Required to obtain collected statistics through the statistics characteristics of the link.

7.6. Component methods

Component methods are called on a component object. They are used to obtain information on the status of the object or to cause the component to undergo some transformation. The following component methods are defined:

Variable methods

`Statistics([Keys() as Integer])`
Read-only property (Object). Required to obtain collected statistics of a global or local variable through Statistical parameter functions. The optional argument **Keys** represents a single string, or a comma-separated sequence of integer key values of the variable data. Note that square bracket pairs [ ] are used instead of parentheses to specify keys in the Python scripting language.

Attribute methods

`Statistics([Keys() as Integer])`
Read-only property (Object). Required to obtain collected statistics for an entity attribute object through Statistical parameter functions. The optional argument **Keys** represents a single string, or a comma-separated sequence of integer key values of the attribute data. Note that square bracket pairs [ ] are used instead of parentheses to specify keys in the Python scripting language. The method call is not effective if multiple replications have been selected and the involved entity has not been assigned an Identifier property.

Distribution methods

`Index`
Function (Integer). Returns the index property.

`Name`
Function (String). Returns the name of the object, which is the Type name concatenated with the Index value.

`Evaluate`
Function (Number). Returns a random number generated by the distribution component.

`Parameters( Index As Integer)`
Property (Floating-point). Returns or assigns the distribution component parameter specified by the argument **Index**. The Index argument is an integer number that represents the sequence number of the parameter, in the order given in the column labeled *Parameters* of the probability density list, starting with the value 1. The permitted parameter value ranges of the available distribution functions are also indicated in the list. Not available for the distribution type *Piecewise linear*.

The index parameter of this property is not used in the Python script language. Instead, all parameters of the distribution component are assigned and retrieved as a tuple.
Example:

```
NormalDistr1.Parameters = (1, 0.01)          (Python)
NormalDistr1.Parameters(1) = 1
NormalDistr1.Parameters(2) = 0.01            (BASIC)
```

Schedule methods

`Index`
Function (Integer). Returns the index property.

`Name`
Function (String). Returns the name of the component, which is the Type name concatenated with the Index value.

`Run( [StreamID As String])`
Function (String). Activates a schedule component. The function works in conjunction with the below-mentioned `Terminate` method. Its behavior depends on the selected value for the Method property of the schedule component. The function call starts a new recurrence stream of schedule events for a schedule of Method *Simulation time*, which will operate concurrent with and independent of other recurrence streams of the schedule. For the schedule method *Clock time* the function reactivates the schedule when inactive. The method is ineffective for the schedule method *Runtime transition*. The **StreamID** argument specifies an identifier for the schedule recurrence stream being created for the schedule Method *Simulation time*. This identifier string is to be matched by the argument passed to the below-mentioned `Terminate` function. A default identifier is assigned and returned if the StreamID argument is omitted. The identifier can only be an empty sting for the schedule method *Clock time.* The call raises a runtime error if the schedule has an active recurrence stream with identifier StreamID.

`Terminate( [StreamID As String])`
Function (Boolean). Deactivates a schedule event recurrence stream for the schedule component. The optional argument **StreamID** is the identifier of the recurrence stream being terminated. An empty string for the StreamID argument targets the Autostart stream. Other values for StreamID are obtained from the above-mentioned Run method. The function returns the value *True* if termination is successful. The function call raises a runtime error if the StreamID argument does not represent the identifier of an active recurrence stream of the schedule.

`Priority`
Function (Integer). Returns the priority value of the schedule component.

Picture methods

`Index`
Function (Integer). Returns the index property of the picture component.

`Name`
Function (String). Returns the name of the picture component, which is the Type name concatenated with the Index value.

7.7. Statistics methods

Renque allows statistics collection for viewport objects and for variable and attribute components. Collected statistics are accessed by a chain of dot-separated method names, starting with the method name `Statistics`.

7.7.1. Statistics methods for viewport objects

The statistics of server and link objects are accessed through:

- The server `Statistics` method
- The link `Statistics` method

The function call raises an error if the object Collect statistics option has not been activated for the server or link object. The `Statistics` method of viewport objects combine in dot notation with one of the following statistics characteristics:

Statistics characteristics methods

`Residence; Population; Idle; Busy`
Read-only property (Object). Provides access to collected statistics for the characteristic that corresponds to the method name. These methods are used only with the above-mentioned `Statistics` method of viewport objects. They combine in dot notation with Statistical parameter functions.

Example:
The expression

```
Srv.Statistics.Residence.Count()
```

returns the number of collected observations for the *Residence* statistic of the server named *Srv*. The parenthesis pair applied to the Count function can be omitted in Basic.

`StatusIdleToBusy`
Assigns the *Busy* status to an object in the *Idle* status. The object will not assume the *Idle* status again until the last resident has departed.

Example:
The expression

```
Srv.Statistics. StatusIdleToBusy()
```

puts the server named *Srv* into the *Busy* state. The parenthesis pair can be omitted in Basic.

7.7.2. Statistics methods for components

Variable statistics are accessed through the variable function:

- The variable `Statistics` method
- The attribute `Statistics` method

These functions raise an error if the Collect statistics option has not been activated for the component. The `statistics` method of components combines in dot notation with the below-mentioned collection of statistical parameter functions.

Examples:

- The expression `Vrb.Statistics(1).Mean()` returns the mean of the global values with key *1* collected for the variable component named *Vrb.* Note that key values are specified using straight brackets in the Python script language as opposed to parentheses for the Basic example.

- The expression: `Srv.Residents(2).Atb.Statistics.Mean(True)` returns the mean of all collected observations for the value of the attribute named *Atb* of the second resident of the server named *Srv*, including the numeric current value.

Note that attribute and variable key values for statistics retrieval are expressed with the Statistic function, as opposed to the component reference itself for accessing component values.

7.7.3. Statistical parameter functions

`Current, Count; Sum; Mean; StdDev; Minimum; Maximum; Rated [IncludeCurrent as Boolean]`
Function (Floating-point | Nil). Returns the value of the corresponding parameter for collected statistics. The parameter functions are defined only as terminator of a cascade of dot-separated method names which starts with the method name `Statistics`. If the value *True* is passed to the optional argument **IncludeCurrent** the function returns the parameter evaluated with the Current property as additional observation. The argument is unavailable for the Current parameter and ineffective for the Residence property. The function returns the value *Nil* for functions other than `Count` if no numerical value can be returned. Some coding examples for these functions are provided in the previous sections.

7.8. Object referencing functions

The objects in a Renque project may be indirectly referenced by class methods, which are bound to a class rather than an instance of the class. There is a class method that finds objects by name and another method that iterates through all existing instances of a class in the project. These methods are available for the following classes:

- `Server`
- `Link`
- `Picture`
- `Schedule`
- `Distribution`

Note that attributes and variable objects cannot be retrieved using class methods.

`Names(TypeName as String, [Index as Integer])`
Function(Object). Returns the object of which the Name property matches the string value passed for the argument TypeName concatenated with the non-default value for the Index argument. The function returns Nil if the specified argument values do not represent an existing object.

Example:
`Server.Names("Srv", 1)`
returns the server object named *Srv1* if existing

`GetNextItem([IteratorReset as Boolean])`
Function (Object). Returns the next object in the set of class instances contained in the project. The function returns the first object if the value *True* is passed for the optional **IteratorReset** argument. No assumptions should be made with regard to the order of the objects returned by the function. The function returns the value *Nil* after the last server object has been addressed. The iterator is reset automatically for all classes at the start of a new replication.

Example:
The following BASIC example script iterates through all server objects in the project:

```
Dim S as Server = Server.GetNextItem(True)
Do Until S = Nil
   ' do something with S
   S = Server.GetNextItem
Loop
```

8. Simulation mechanism

This section focuses on the architecture of the Renque simulation engine. The principal constituents of a simulation model are server and link objects. A Renque discrete event simulation essentially encompasses the distribution of entities across servers and links. Server objects drive the simulation by entity retention and routing. Link objects connect the servers and accommodate transport of entities between servers by travel. Links also create entities.

8.1.1. Link operation mechanism

Entities created on or relocated to a link are conveyed on the link by travel. Links have a fixed travel direction. Travel is instantaneous and terminated by the Arrival calendar event, which signals the destination server. The entity is stored as resident in the link if the destination server fails to collect the entity in response to the arrival signal. If the link is a downstream open link (with no destination server) entities are destructed after travel without arrival signal.

8.1.2. Server operation mechanism

Servers can store entities as resident in the server. The processing procedure of server residents is referred to as ***routing***. Entity routing is accomplished either by the server script procedure `ArrivalAccept` or by the entity script procedure `Route`.

Routed entities can be stored for a specific time period, or without time limit. When the applied storage time has passed a routing event removes the entity from the server and sends it to a preassigned downstream connected link. If no recipient link was assigned the entity is destructed. The latter is also referred to as ***expiration***. A server can store an unlimited number of residents simultaneously, each with their own routing properties.

Block and fusion servers act as container for other servers and links. They have independent data recording and statistics collection. The simulation behavior of a server or a link is not affected by the condition of being contained in a block or fusion server.

8.1.3. Default customizations

All server or link objects have a default customization, assigned upon creation of the object. The default link customization script, executed once at the start of a simulation replication, is given by:

```
if .Origin = Nil and .Destination <> Nil then .EntityCreate
```

The default server customization script is given by:

```
if Not .ArrivalAccept then .EntityProcure
```

The preprogrammed default link customization script causes the initial creation of an entity on upstream open links. This default server customization is aimed at continuation of the flow of entities through a server, if accommodated by the applied server properties Delay and Signal departures.

8.1.4. Component basics

Components are user-creatable project elements which are not displayed on the viewport. Components are created in the Model console of the configuration window. The are four component types:

- Variable — Allows storage of data as global property, or as property of a server or link object.
- Attribute — Allows storage of data in entities.
- Distribution — Generates random numbers in accordance with a specified probability density distribution.
- Schedule — Permits execution of a customization script at specific time points in a simulation.

8.1.5. Signaling

Most practical simulation models make extensive use of customization. The customization scripts of server and link objects are executed in response to signals triggered by certain simulation incidents.

Signal triggers

The only signal type for link objects is the **initialization** signal. All links receive the initialization signal once, at the start of a simulation replication. New links introduced into the model in the pause state receive the initialization signal when the simulation run is resumed.

Server objects have three signal types:

- **Arrival** — Triggered by the arrival event for completed entity travel on an upstream connected link.
- **Departure** — Triggered by a Routing or Expiration event that was created under application of the Signal departures server property. This signal is received before the departing resident entity is removed from the server.
- **Scripted** — Triggered by the signal event, which is created by the `SignalSend` server script command.

In addition, a Schedule customization is executed by a schedule event.

Signal properties

A signal has several properties that reflect the conditions of the signal. These properties are accessible by scripting through signal methods.

- `Entity` — Represents the entity object involved with the **Arrival** and **Departure** server signals. The property is *Nil* for all other signal types.
- `LinkPosition` — Represents the connection order of a specific link involved with a server signal. The LinkPosition property of the **Departure signal** represents the target link of the entity, with value *0* for destructed entities. For the **Arrival signal** the LinkPosition property identifies the link that carries the entity to the server. For the **Scripted signal** the property is assigned by the `SignalSend` script command.
- `Recipient` — Link or server object of a signal for an object customization, or the schedule component of a schedule customization.

Furthermore, the signal method `IsArrival` can be used to verify whether or not the customization is executed in response to a server arrival signal. The server signal type can also be determined by the signal properties, as summarized in the table given below.

Property \ Type	**Arrival**	**Departure**	**Scripted**
Has signal entity	Yes	Yes	No
LinkPosition	< 0	>= 0	Any

8.1.6. Event calendar

The heart of the simulation engine is the event calendar. The event calendar is a list of successively executed calendar events. All simulation actions result from execution of calendar events. A calendar event is defined by an event type, an event time, a priority value and usually a simulation object and an entity. The event object can be a server, a link, or a schedule component.

A Renque simulation is performed by consecutive execution of calendar events. At the start of a simulation replication a number of initialization events are created in the event calendar. Further addition of calendar events occurs by calendar event actions only. When the event calendar is empty the simulation terminates.

The events are executed in order of event time. The event time of a newly created calendar event is always greater than or equal to the event time of the currently executed event.

Event priority

Events with identical event time are executed in order of Event priority. The execution order of events with both identical event time and priority value is the order of event creation. The priority of an event is an integer number in the range -32765 to +32765. The value is determined primarily by the Priority property of servers and the Priority property of schedule components, as discussed in the next topic.

Event types

The table below lists all calendar event types with a concise description of their purpose.

Event type	Priority	Description
Prime	System	A simulation replication starts with a system ***Initialization*** event. This event creates initialization events for links. The event also creates ***Schedule*** events for Autostarting schedule components, and an initializing ***Clock*** event.
Initialization	Dest. server, System	Triggers the link initialization signal, which executes the link customization.
Arrival	Origin server, Custom	Handles the arrival of an entity at the downstream side of a link by travel.
Routing	Host server	Removes a resident entity from the server for routing to a predetermined downstream link.
Expiration	Host server	Routing event without recipient link. Removes a resident entity from the server and destructs the entity.
Signal	Server	Triggers the scripted server signal in response to the `SignalSend` command, which executes the customization script of the server.
Schedule	Schedule	Executes the customization script of a schedule component and creates a new Schedule event for the next schedule timing recurrence.
Clock	System	Controls synchronization of the Calendar-based clock program by activation of a timetable or termination of the replication on the specified End date.

The first column of the table specifies the name of the event type, as displayed in the Event viewer. The second column indicates the evaluation method of the event priority for the event type. The phrase *System* refers to the so-called system priority, which is the highest possible priority value for an event. The phrases *Server* and *Schedule* indicate that the event receives the Priority value of the server that causes creation of the event or the Priority value of the schedule component. The phrase *Custom* indicates that the event priority is determined by the script commands `EntityCreate` and `Transfer`, both of which are not generally associated with a particular server.

Simulation timing

The simulation time is the principal time parameter of a simulation. The simulation time has the value zero when a simulation starts. Because of the sequential execution of calendar events the simulation time does not proceed continuously as the real time does, but jumps to the event time of the next event when it is executed. Renque also has a user-adaptable clock that allows time-related properties to be specified and expressed in the reference frame of the 12 or 24-hour UTC time and the Gregorian calendar. Customization clock methods, time display strings and schedule components all have access to the simulation time and the simulation clock.

8.1.7. Server and link status

The two possible status values of a server are *Idle* and *Busy*. The server status is *Idle* when no entities are stored as resident, *Busy* when it has at least one resident. The transition from idle to busy can also be accomplished by the script command `StatusIdleToBusy`.

8.1.8. Random number generation

Simulation results may possess stochastic properties as a result of the application of distribution components in the project. These components make use of a built-in pseudo random number generator, which is based on the widely used *Xoroshiro128+* algorithm. The random number generator produces a sequence of floating-point numbers $0 < x_i < 1$, starting from a specified seed value. The sequence is endless for any practical objective. Because the seed value is coded into Renque, the random number sequence is reproduced for every simulation on any system.

9. Tools & utilities

The App has a variety of utilities that assist the user in project construction and review. The most commonly used tools are:

- Chart configuration tool
- Object alignment utility
- Search utility
- Window arrangement utility
- Script editor
- Picture management
- Preferences

9.1. Chart configuration

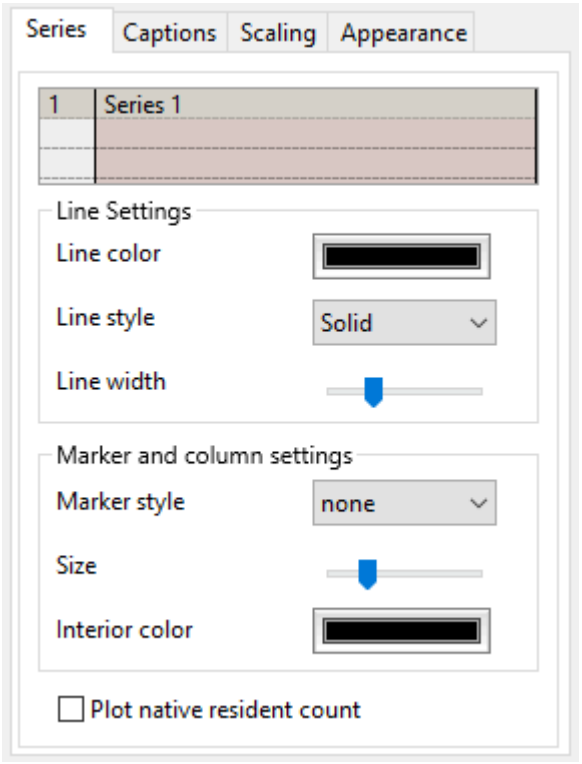
This chapter discusses configuration properties of chart properties which affect chart display on the viewport.

Chart properties are available when the value *Chart* has been selected for the Shape server property. The **Config** button adjacent to the Shape drop-down control toggles the display of **chart properties pane**. Chart data can be displayed in columns and in xy-scatter or line plots. Chart data are controlled through a set of Renque Chart script commands.

The display properties of a chart are controlled on the chart panel. The chart panel has four tab-pages, each presenting a specific category of chart elements.

9.1.1. Series

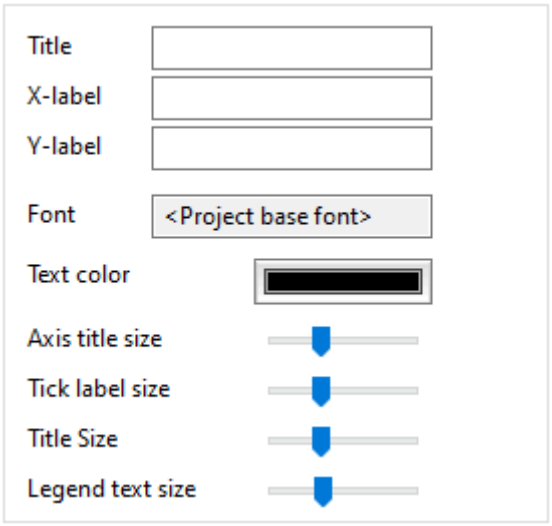
The Series tab controls the styling attributes of the individual data series of the chart. The series list at the top allows creation, deletion, renaming, reordering and selection of a series item.



The properties of a single selected series are managed by the controls in the **Line settings** frame and the **Marker and column settings** frame, which control such properties as line width and the marker symbol used to indicate data points. The **Plot native resident count** checkbox engages plotting of the resident count of the chart server as a function of the simulation time in an automatically generated data series.

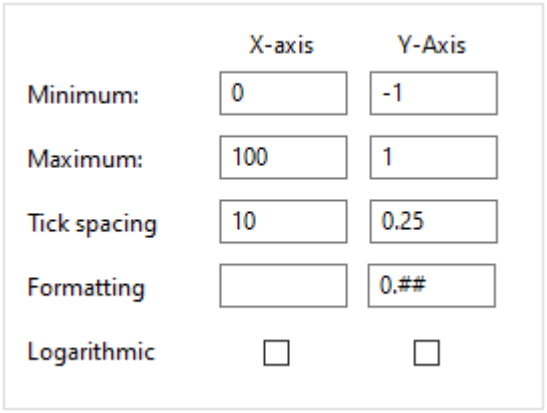
9.1.2. Captions

The Captions tab controls the font, size and color of text displayed on the chart, as indicated by the labels on the tab.



9.1.3. Scaling

The Scaling tab controls scale ranges of the horizontal and vertical axes of the chart

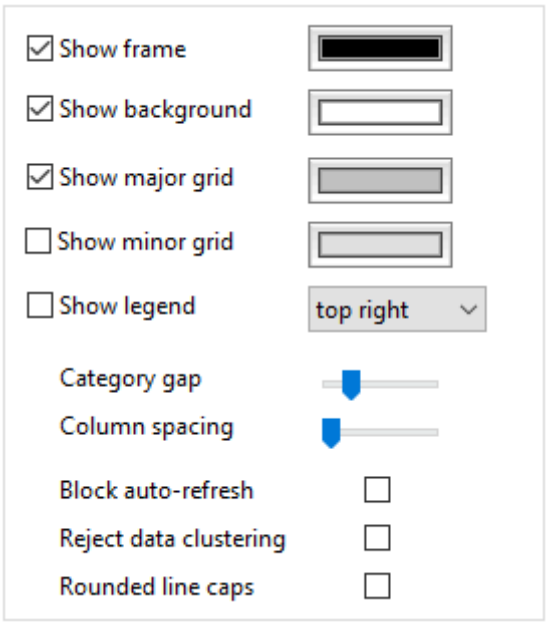


The text boxes labeled **Formatting** determine the format of the tick labels, specified by Number format characters as also used for clock time display format. The **Logarithmic** checkboxes independently engages a log scale for both axes. The x-axis scale range of line charts can be focused dynamically by rotating the **mouse wheel** while hoovering over the horizontal chart axis in the viewport with the control and shift key both pressed.

9.1.4. Appearance

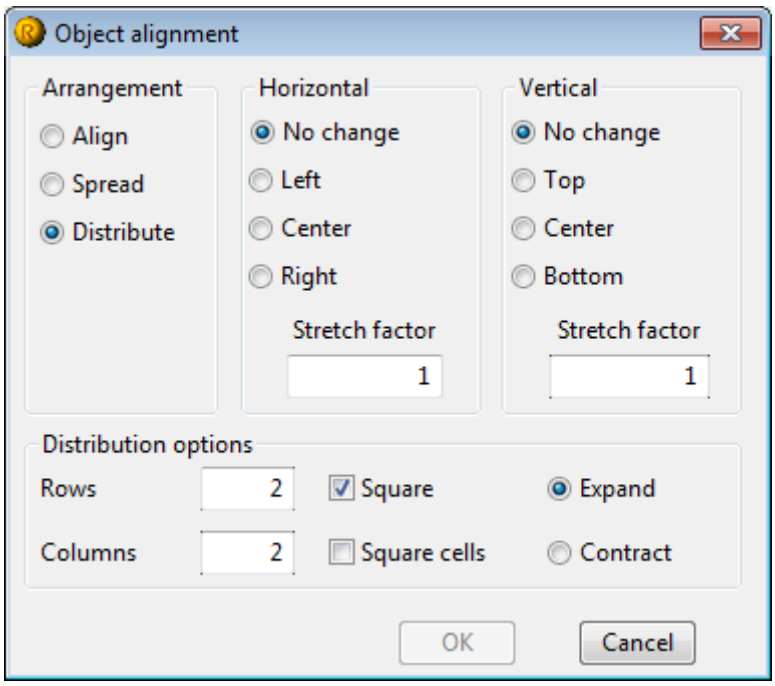
The Appearance tab controls a choice of additional decoration elements and style features of the chart. The check boxes are used to toggle the visibility of specific chart elements. The **Category gap** and **Column spacing** sliders allow adjustment of column width and spacing in column charts. The **Block autorefresh** check box

exempts the chart from updates by regular animation update intervals. The **Reject data clustering** check box disables data compression for accelerated plotting of large data sets. The **Rounded line caps** check box affects the line cap smoothness for thick line styles in line plots.



9.2. Object alignment utility

The alignment utility allows alignment of the position of servers on the Viewport. It is activated by clicking the **More...** item of the Align submenu of the Format menu. Alignment only affects servers selected in the Viewport. While the alignment utility is activated the Workspace window and all Fusion windows are disabled until the alignment utility is closed.



The frames labeled Horizontal and Vertical determine the alignment orientation. The options in the **Arrangement** frame determine the alignment method. The default *Align* option allows alignment of the server object by the options selected in the adjacent **Horizontal** and Vertical frames. Selection of the *Spread* option allows scaling of the server arrangement by a factor specified in the **Stretch factor** text fields. The *Distribute* option allows rearrangement organized in rows and columns in the configuration specified by the items in the Distribution options frame.

No change

No changes are made to the server positions in either the horizontal or vertical direction.

Left, Center, Right

Aligns the horizontal position of respectively the left-most edges, the center and right-most edges of the selected server images to the server object last selected. If all objects were selected by a rectangular sweep the alignment reference is the horizontal center of the selection.

Top, Center, Bottom

Aligns the vertical position of respectively the top-most edges, the center and bottom-most edges of the selected server images to the server object last selected. If all objects were selected by a rectangular sweep the alignment reference is the vertical center of the selection.

Stretch factor

Scales the realignment by horizontal and vertical scaling factors

Row count
Number of rows formed by the alignment.

Column count
Number of columns formed by the alignment.

Square
Rearranges the server objects into array with square outline.

Square cells
Rearranges the server objects into an array with square cells.

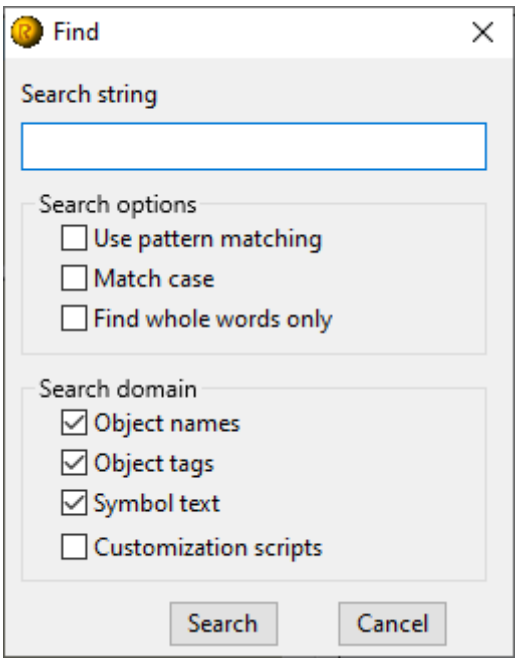
Expand
Increases the width or height of the selection span to obtain square or square cell distribution.

Contract
Decreases the width or height of the selection span to obtain square or square cell distribution.

9.3. Search utility

The search utility can be used to search for objects in the viewport. It is activated by the **Find** item of the Edit menu, or by pressing the shortcut key combination **<ctrl>F** in the Viewport. The utility can be opened from the Workspace window and from a Fusion window. The search domain includes only objects in visible layers of the viewport of the window from which the utility was opened. The utility searches for a user-specified string. All objects that satisfy the search criterion are selected in the Viewport.

When the utility is activated the Find window appears and the Main window and all Fusion windows are disabled until the search utility is closed. The utility window has a **Search text** field, where the search string is entered and two option frames which determine the search domain and method.



9.3.1. Search options

Use pattern matching
Enables pattern matching using Regular expressions.

Match case
Enables case sensitive searching.

Find whole words only
Restricts the search to whole words.

9.3.2. Search domain

Object names
Searches the Name property of server and link objects.

Object tags
Searches in Tags of server and link objects.

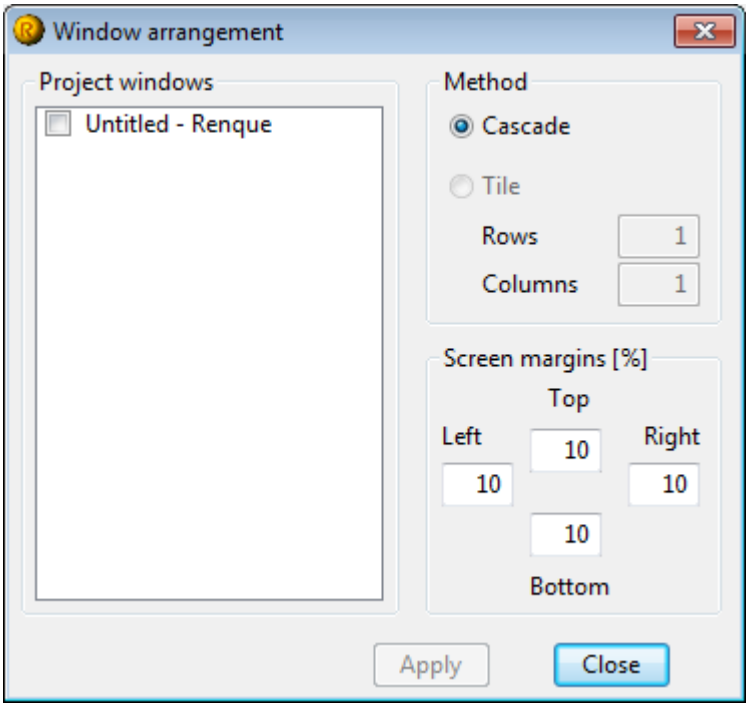
Symbol text
Searches the Text property of server symbol objects.

Customizations
Searches the script code of server and link customization scripts.

9.4. Window arrangement utility

The window arrangement utility can be used to arrange the various windows opened by the app on the monitor screen. It is activated by the **Arrange Windows** item of the Window menu.

The utility window contains three frames. The Method frame determines the way the window rearrangement is carried out. The frame labeled *Windows* controls which windows are rearranged and determines the repositioning order of the windows. The Screen margins frame defines a screen area for placement of the rearranged windows. On a system with multiple monitors the windows are placed on the monitor on which the utility window is displayed.



Project windows

The list in the Project windows frame displays the name of each window of the project that may be subjected to rearrangement. The order of the listed items can be modified by drag-drop operations. The arrangement is performed for all items checked in the list.

Arrangement method

Defines the arrangement method. Application of the **Cascade** option stacks the windows in a such manner that overlap of the title bar of the rearranged windows is minimized. The **Tile** option arranges the windows in as many **Rows** and **Columns** as specified in the two associated text fields, in such a way that no window overlaps any other window. For both options the windows are uniformly resized.

Screen margins [%]

Defines the margins on all four edges of the target area in percentages of the screen height or width.

9.5. Script editor

The script editor is used to compose customization scripts for viewport objects and for schedule components.

The text area of the editor frame is a basic text edit area of the operating system with some additional features that assist script coding.

9.5.1. Coding aids

Automatic code completion

This feature automatically fills out a script method and object names while editing scripts. A list of selectable phrases appears in a popup list on the text area when the **Tab key** (Basic script language only) or **Ctrl+Space key** is pressed. The list contains method names which represent an appropriate continuation of the text preceding the caret position on the text area. The list contains object and component names if the Tab key or spacebar is pressed while holding down the **Shift key**. The desired item in the list can be selected with the navigation keys on the keyboard. Pressing the **Tab key, spacebar** or the **Enter key** will complete the text at the caret position with the selected item in the pop-up list. Pressing the **Esc key** closes the list without making any changes to the script code.

Method syntax help

Renque provides information on the correct syntax of script methods in the non-editable help pane at the bottom of the text area. If the mouse hovers over a method or object name the help pane shows information on the syntax such as command argument name and type, function return value type and object type.

Comment button

The comment button on the top right of the frame converts selected code lines into comments by inserting a hyphen character at the beginning of the line. If all selected lines start with a hyphen character the hyphens are removed in order to convert the comments back into script code.

Script compiler test

The compiler button at the top left indicates the script language selected for the present customization by an icon. The script language selection may be altered by clicking on this button with the right mouse button. A mouse click on this button performs precompiles the script. The number of resulting compiler errors is given in the label adjacent to the test button. Each error is highlighted in the text area with a distinct background color. Pressing the **Ctrl-G** shortcut key combination selects the

text of the next error in the text area, or the previous in combination with the Shift key. The help pane at the bottom provides an error description when the mouse hovers over an error highlight in the text area.

9.6. Picture management

The picture management console is activated by the **Pictures** command of the Tools menu. The picture repository contains pictures that can be used to represent servers in the viewport. The console has a picture repository to the left and a tab strip on the top right to access different picture property categories for pictures selected in the repository. The picture repository provides a list of the pictures in the project. Pictures can be added from disk by the **New** command of the contextual menu of the repository, by dropping a picture file on the repository, or by pasting a bitmap from the clipboard into the repository. The picture viewer on the bottom right displays the bitmap of the pictures selected in picture repository organized in a grid. A click on a picture in the picture viewer deselects all other pictures.

9.6.1. Masking

Transparency color

Defines the transparency mask color of the selected picture. The transparency color can be selected by the color selection dialog activated by a click on the adjacent color display field. The transparency color can also be selected by a click on a pixel in the picture viewer, provided a single picture is selected in the picture repository. The transparency mask color is opaque and does not preserve any transparency information of the selected pixel.

Layer substitution color

Defines the mask color of the selected picture to be substituted with the color of the object layer. The layer color can be selected by the color selection dialog activated by a click on the adjacent color display field. The layer color can also be selected by a click on a pixel in the picture viewer while pressing the **Shift key**, provided a single picture is selected in the picture repository. The mask color is opaque and does not preserve any transparency information of the selected pixel.

9.6.2. Knots

Add knot

Activates knot creating. Picture knots can be used as anchor for attachment of a link to a server represented by the picture, with the Icon fit value *Filled fit* selected. A click on the picture viewer with activated knot creating results into creation of a new knot in the picture on the mouse position, with the knot location indicated in pixels in the text fields labeled X and Y.

Required link snap opacity

Specifies the minimum opacity level percentage required for attachment of a link to a server represented by the picture.

9.6.3. Transform

Crop transparent edges

Crops the picture by removing transparent edges on all four sides of the items selected in the picture repository.

Replace

Activates a file browser to select a picture file for replacement of the items selected in the picture repository.

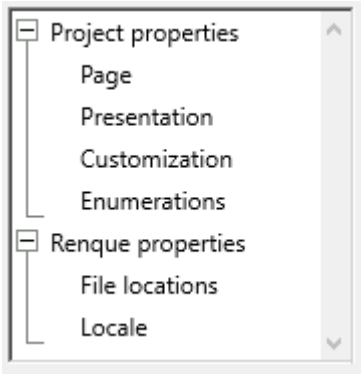
9.6.4. Animated

Entity icon scaling

Activates scaling of entity icons with the viewport scaling factor for entities represented by the picture object. If unchecked (default) the picture is displayed in the viewport at the picture bitmap size for entity icons, independent of the applied viewport scale.

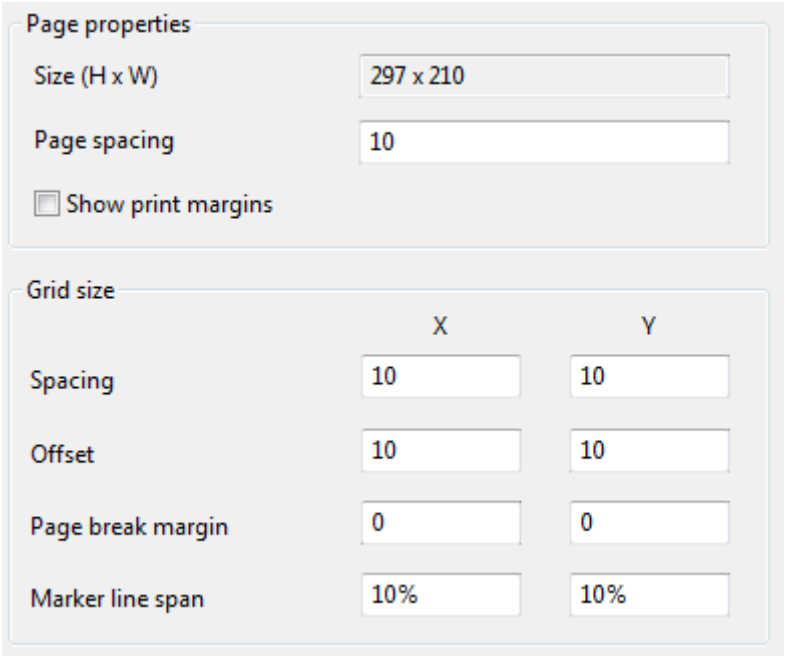
9.7. Preferences

The Preferences console is activated by the **Preferences** command of the Tools menu. Selecting an item in the category list at the left side of the console presents the corresponding properties pane at right.



9.7.1. Page layout

The page properties pane determines the page properties of the project.



Size (H x W)

Displays the page size expressed in the selected for the Project base unit.

Page spacing

Determines the page distance on the viewport in the value selected for the Project base unit.

Show print margins

If checked, the page margins of the selected printer are displayed by a line in the viewport.

9.7.2. Grid size

Spacing
Determines the distance between the grid lines in the viewport specified in the value selected for the Project base unit.

Offset
Determines the distance of the top and left grid line to the page edge specified in the value selected for the Project base unit.

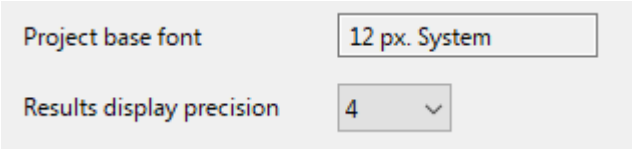
Page break margin
Determines the required minimum distance to the right and bottom page edges for gridlines specified in the value selected for the Project base unit.

Marker line span
Determines the display length of the gridlines as percentage of the spacing.

Project base unit
Determines the base unit for size display and specification.

9.7.3. Presentation

Determines the display properties of the viewport.



Project base font
Determines the default font for text in the viewport. A click on the text field opens the font picker tool, by which the default font can be changed,

Results display precision
Determines the number of significant digits for display of numerical results.

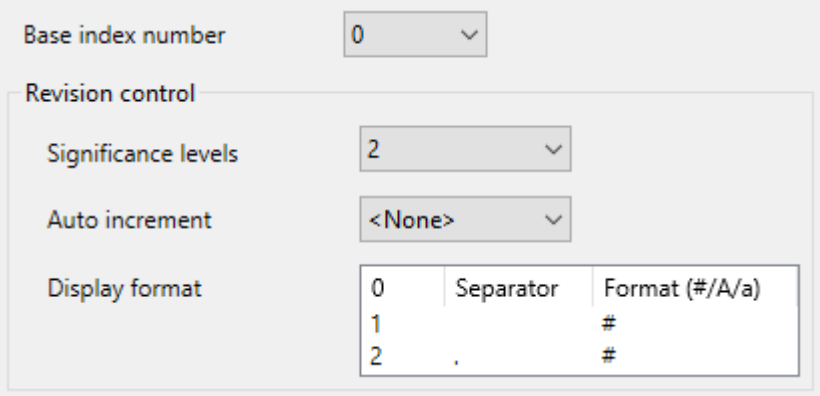
9.7.4. Customization

Determines the script language and script language properties.

Scripting language
Selects the script language for server, link or schedule objects for which no customization script (other than the default script) has yet been created.

Script timeout
Determines a maximum run time for a single customization in seconds real time for the Python language. When this value is exceeded, a runtime error is raised and execution of the script terminates.

9.7.5. Enumerations



Base index number

Determines the lowest index number assigned to new link and server objects. The dropdown menu allows selection of the values *0* or *1*.

Revision control

The revision control frame determines the method by which version identification is organized.

Significance levels

Determines the revision enumeration depth. Revision enumeration is applied to distinguish significance levels of revisions. The lowest revision significance level can be updated automatically.

Auto increment

Determines the method for automated increment of the lowest significance enumeration. The dropdown menu allows selection of the values *<None>*, *On save* and *On exit*. The value *<None>* requires manual revision assignment by the contextual menu of the Revision text field.

Display format

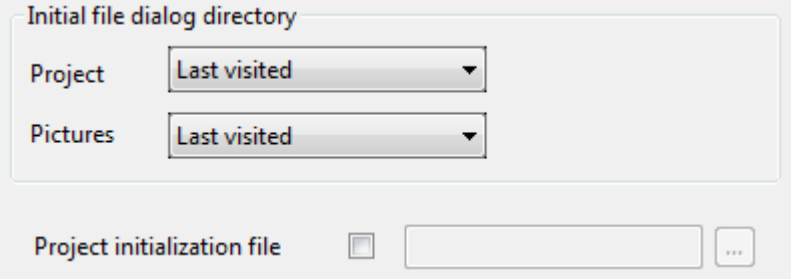
Determines the revision display style in the Revision text field and by Viewport object text codes. The string specified in the column labeled *Separator* is shown between the significance level values. The *Format* column requires a selection out of the three characters to specify the revision display format. A series of hashtag characters (*#)* represent the minimum number of digits displayed for the integer number. The value *A* presents uppercase letters in the scheme A, B, C ... AA, AB... The value *a* presents lowercase letters by the same scheme. If no format is specified the revision enumeration is displayed as unformatted number.

9.7.6. File locations

Initial file dialog directory

Project

Determines the folder for the **Open** and **Save** commands of the File menu.



The three options are: The *Last visited* folder by file operations within the app, the *Documents folder* of the operating system, and a *Specified location*, which is assigned in the adjacent text field that appears when selected.

Pictures

Determines the browsing folder for new pictures created by the Picture management utility.

Project initialization file

Enables loading a project file with every new project. The file and path name are shown in the adjacent text field. The initialization file is selected using the browse button to the right of the text field.

9.7.7. Locale

Units system

Determines the units of measurement preference for the application. The dropdown menu has the values *Metric*, *Imperial* and *None.* The selection affects the default value for the Project base unit and evaluation of standard printer paper dimensions. The values *Metric* or *Imperial* are recommended if high accuracy page size representation is expected for a matching page setup selection.

10. Appendices

This section provides general information associated with the Renque software application for reference.

- Regular expressions
- Text markup
- Script method name list

10.1. Regular expressions

This section describes the syntax of regular expressions.

Pattern	Description
.	Matches any character except newline.
[a-z0-9]	Matches any single character of set.
[^a-z0-9]	Matches any single character not in set.
\d	Matches a digit. Same as [0-9].
\D	Matches a non-digit. Same as [^0-9].
\w	Matches an alphanumeric (word) character - [a-zA-Z0-9_].
\W	Matches a non-word character [^a-zA-Z0-9_].
\s	Matches a whitespace character (space, tab, newline, etc.).
\S	Matches a non-whitespace character.
\n	Matches a newline (line feed).
\r	Matches a return.
\t	Matches a tab.
\f	Matches a form feed.
\b	Matches a backspace.
\0	Matches a null character.
\000	Also matches a null character because of the following:
\	Matches an ASCII character of that octal value.
\x*nn*	Matches an ASCII character of that hexadecimal value.
\c*X*	Matches an ASCII control character.
metachar	Matches the meta-character (e.g., \, .).
(abc)	Used to create subexpressions. Remembers the match for later backreferences. Referenced by replacement patterns that use \1, \2, etc.
\1, \2,…	Matches whatever first (second, and so on) of parens matched.
x?	Matches 0 or 1 **x'**s, where x is any of above.
x*	Matches 0 or more **x'**s.
x+	Matches 1 or more **x'**s.
x{**m**,**n**}	Matches at least **m x'**s, but no more than **n**.
abc	Matches all of a, b, and c in order.
a|b|c	Matches one of a, b, or c.
\b	Matches a word boundary (outside [] only).

\B	Matches a non-word boundary.
^	Anchors match to the beginning of a line or string.
$	Anchors match to the end of a line or string.

10.1.1. Replacement Patterns

The following expressions can only apply to the replacement pattern:

Pattern	Description
$'	Replaced with the entire target string before match.
$&	The entire matched area; this is identical to \0 and $0.
$'	Replaced with the entire target string following the matched text.
$0-$50	$0-$50 evaluate to nothing if the subexpression corresponding to the number doesn't exist.
\0-\50	
\x*nn*	Replaced with the character represented by *nn* in Hex, e.g., ™is ™.
\n*nn*	Replaced with the character represented by *nn* in Octal.
\c*X*	Replaced with the character that is the control version of *X*, e.g., \cP is DLE, data line escape.

10.1.2. Regular Expression Examples

The basic idea of regular expressions is that it enables you to find and replace text that matches the set of conditions you specify. It extends normal Search and Replace with pattern searching.

Wildcards

Some special characters are used to match a class of characters:

Wildcard	Matches
.	Any single character except a line break, including a space.

If you use the "." as the search pattern, you will select the first character in the target string and, if you repeat the search, you will find each successive character, except for Return characters.
The following wildcards match by position in a line:

Wildcard	Matches	Example
^	Beginning of a line (unless used in a character class; see below)	^Phone: Finds lines that begin with "Phone":
$	End of a line (unless used in a character class)	$: Finds the last character in the current line.

Character Classes

A character class allows you to specify a set or range of characters. You can choose to either match or ignore the character class. The set of characters is enclosed in

brackets. If you want to ignore the character class instead of match it, precede it by a caret (^). Here are some examples:

Character Class	Matches
[aeiou]	Any one of the characters a, e, i, o, u.
[^aeiou]	Any character except a, e, i, o, u.
[a-e]	Any character in the range a-e, inclusive
[a-zA-Z0-9]	Any alphanumeric character.
[[]	Finds a [.
[]]	Finds a]. To find a closing bracket, place it immediately after the opening bracket.
[a-e^]	Finds a character in the range a-e or the caret character. To find the caret character, place it anywhere except as the first character after the opening bracket.
[a-c-]	Finds a character in the range a-c or the - sign. To match a -, place it at the beginning or end of the set.

Non-printing Characters

You can use the following notation to find non-printing characters:

Special Character	Matches
\r	Line break (return)
\n	Newline (line feed)
\t	Tab
\f	Formfeed (page break)
\x*NN*	Hex code **NN**.

Other Special Characters

The following patterns are wildcards for the following special characters:

Special Character	Matches
\s	Any whitespace character (space, tab, return, linefeed, form feed)
\S	Any non-whitespace character.
\w	Any "word" character (a-z, A-Z, 0-9, and _)
\W	Any "non-word" character (All characters not included by \w).
\d	Any digit [0-9].
\D	Any non-digit character.

Repetition Characters

Repetition characters are modifiers that allow you to repeat a specified pattern.

Repetition Character	Matches	Examples
*	Zero or more characters.	d* finds no characters, or one or more consecutive "d"s. .* finds an entire line of text, up to but not including the return character.
+	One or more characters.	d+ finds one or more consecutive "d"s. [0-9]+ finds a string of one or more consecutive numbers, such as "90404", "1938", the "32" in "Win32", etc.
?	Zero or one characters.	d? finds no characters or one "d".

Please note that, since * and ? match zero instances of the pattern, they always succeed but may not select any text. You can use them to specify an optional character, as in the examples in the following section.

"Greediness"

The "?" is used as a "greediness" modifier for a subpattern in a regular expression. You can place a "?" directly after a * or + to reverse the "greediness" setting. That is, if Greedy is True, using the ? after a * or + causes it to match the minimum number of times possible: For example, consider the following.

Target String	Greedy	Regular Expression	Result
aaaa	True	(a+?) (a+)	$1=a, $2=aaa
aaaa	False	(a+?) (a+)	$1=aaa, $2=a

Extension Mechanism

We also support the regular expression extension mechanism used in Perl. For instance:

(?#text)	Comment
(?:pattern)	For grouping without creating backreferences
(?=pattern)	A zero-width positive look-ahead assertion. For example, \w+(?=\t) matches a word followed by a tab, without including the tab in $&.
(?!pattern)	A zero-width negative look-ahead assertion. For example foo(?!bar)/matches any occurrence of "foo" that isn't followed by

	"bar".
(?<=pattern)	A zero-width positive look-behind assertion. For example, (?<=\t)\w+ matches a word that follows a tab, without including the tab in $&. Works only for fixed-width look-behind.
(?<!pattern)	A zero-width negative look-behind assertion. For example (?<!bar)foo matches any occurrence of "foo" that does not follow "bar". Works only for fixed-width look-behind.

Subexpressions

You can use parentheses within your search patterns to isolate portions of the matched string. You do this when you need to refer to subsections of the matched in your replacement string. For example, you would do this if you need to replace only a portion of the matched string or insert other text into the matched string.

Here is an example. If you want to match any date followed by the letters "B.C." you can use the pattern "\d+\sB\.C\." (Any number of digits followed by a space character, followed by the letters "B.C.") This will match dates such as 33 B.C., 1742 B.C., etc. However, if you wanted your replacement pattern to leave the year alone but replace the letters with something else, you would use parens. The search pattern "(\d+)\s(B\.C\.)" does this.

When you write your replacement pattern, you can refer to the year only with the variable \1 and the letters with \2.

If you write "(\d+)\s(B.C.|A.D.|BC|AD)", then \2 would contain the matched letters.

Combining Patterns

Much of the power of regular expressions comes from combining these elementary patterns to make up complex searches. Here are some examples:

Pattern	Matches
\$?[0-9,]+\.?\d*	Matches dollar amounts with an optional dollar sign.
\d+\sB\.C\.	One or more digits followed by a space, followed by "B.C."

The Alternation Operator

The alternation operator (|) allows you to match any of a number of patterns using the logical "or" operator. Place it between two existing patterns to match either pattern. You can use more than one alternation operator in a pattern:

Pattern	Matches
\she\s | \sshe\s	" he " or " she "
cat|dog|possum	"cat", "dog", or "possum"

([0-9,]+\sB\.C\.)|([0-9,]+\sA\.D\.) "'or'" [0-9,]+\s((B\.C\.)|(A\.D\.))	Years of the form "**yearNum** B.C. **or** A.D." e.g., "2,175 B.C." or "215 A.D."

10.1.3. Search and Replace

You use special patterns to represent the matched pattern. Using replacement patterns, you can append or prepend the matched pattern with other text.

Pattern	Description
	Contains the entire matched pattern.
$&	If "\d\d\d\d\sB\.C\." finds "1541 B.C.", then the replacement pattern "the year $&" results in "the year 1541 B.C.", as the $& contains the string "1541 B.C".
	Contains the matched subpatterns, defined by use of parentheses in the search string.
\1, \2, etc.	The search pattern "(\d+)\s(B\.C\.|A\.D\.|BC|AD)" looks for any number of digits followed by a space character, followed by either "B.C.", "BC", "A.D.", or "AD". The \1 variable contains the match to the "\d+" portion of the expression and \2 contains the match to the "B\.C\.|A\.D\.|BC|AD" portion.

Credits

Philip Hazel, and copyright by the University of Cambridge, England.

10.2. Text markup escape sequences

This section provides a list of escape sequences available for insertion of object or clock properties in tag or symbol text. An escape sequence consists of the backslash escape character followed by two other characters. Two consecutive backslashes represent a single literal backslash.

10.2.1. Escape sequences for object properties

An escape sequence in object property markup is replaced by a string that expresses a property value in the text displayed in the viewport. The following object escape sequences are defined in Renque:

Esc. seq.	***Description***
\OT	Object type name
\OI	Object index
\Oi	Non-zero object index
\OX	Object index, not displayed if type name is unique for the object
\OP	Object icon picture name (server only)
\DT	Object drawing type name.
\DI	Object drawing index
\Di	Non-zero drawing index
\DX	Drawing index, not displayed if type name is unique for the drawing
\DR	Drawing revision
\PN	Project name
\PR	Project revision

Drawing designation
The drawing container of a server object is determined by the center of the image that represents the server in the viewport. The drawing object of a link is determined by the tag seat location of the link. If an object is positioned in the page break margin area between two pages the corresponding drawing escape sequence is undetermined, indicated by the string “<*Property* ref>”

Page reference representations
The lead characters *O* and *D* are case-sensitive for the index referencing sequence of page reference servers. Lower case for the first character denotes the index property of the paired server of a paired page reference server, as opposed to the index property of the server itself.

10.2.2. Escape sequences for clock properties

A clock markup escape sequence is replaced by a string that expresses a clock property value, in the text as displayed in the viewport. The following clock escape sequences are defined in Renque:

General clock display sequences

key	***Description***
\st	Displays the simulation time with no formatting.
\^t	Inserts a tab stop.

Clock display sequences for the unit-based clock

\TL	Clock time in the long date format of the operating system.

\TS Clock time in the short date format of the operating system.
\wd Full weekday name in the English language
\w2 Weekday name abbreviated to two characters
\w3 Weekday name abbreviated to three characters
\sW Week count in the simulation time
\sD Day count in the simulation time
\sH Hour count in the simulation time
\sM Minute count in the simulation time
\sS Second count in the simulation time
\NW Week count in the clock
\ND Day count in the clock
\NH Hour count in the clock
\NM Minute count in the clock
\NS Second count in the clock

Clock display sequences for the calendar-based clock

\CS Clock calendar date in the short date format of the operating system.
\CL Clock calendar date in the long date format of the operating system.
\CY Year number of the current clock calendar date.
\CM Month number of the current clock calendar date.
\CW Week number of the year of the current clock calendar date.
\CD Day number of the month of the current clock calendar date.
\CN Number of days into the year of the current clock calendar date.

The 'count' sequences are case-sensitive for the second character. A capital performs a 1-based count and lower case performs a 0-based count.

Example
Consider a simulation run with the following clock properties:

Calendar-based:	Yes
Simulation start:	Wednesday March 5, 1980
Wednesday timetable:	02.00 to 24.00 hrs.
Time unit:	minutes

At 4:10:20 am on the simulation start date, the following clock representations are obtained:

Default	5-3-1980 4:10:20 (depends on regional settings of the OS)
\st	130.3333
\TS	4:10 (depends on regional settings of the OS)
\sH	3
\NH	5
\Nh	4

An escape sequence that returns a numeric value may have a tailing combination of number formatting characters to format the replacement number. Number formatting characters are given in the following table:

Number format characters

Character	Description
#	Placeholder that displays the digit from the value if it is present. If fewer placeholder characters are used than in the number passed, then the result is rounded.
0	Placeholder that displays the digit from the value if it is present. If no

	digit is present, 0 (zero) is displayed in its place.
.	Placeholder for the position of the decimal point.
'	Placeholder that indicates that the number should be formatted with thousand-separators.
%	Displays the number multiplied by 100.
(	Displays an open paren.
)	Displays a closing paren.
+	Displays the plus sign to the left of the number if the number is positive or a minus sign if the number is negative.
-	Displays a minus sign to the left of the number if the number is negative. There is no effect for positive numbers.
E or e	Displays the number in scientific notation.

The absolute value of the number is displayed. You must use the + or - signs if you want the sign displayed. Although the special formatting characters are U.S. characters, the actual characters that will appear are based on the current operating regional system settings.

The number format sequence can be made up of up to three formats separated by semicolons. The first format is the format to be used for positive numbers. The second format is the format to be used for negative numbers and the third format is the format to be used for zero.

Examples
The following are several examples that use the various special formatting characters.

Format	Number	Formatted String
#.##	1.786	1.79
#.0000	1.3	1.3000
0000	5	0005
#%	0.25	25%
###,###.##	145678.5	145,678.5
#.##e	145678.5	1.46e+5
-#.##	-3.7	-3.7
+#.##	3.7	+3.7

10.2.3. Time unit conversion

Properties referencing the simulation time are expressed in the selected Time unit of the clock. Dimensional time values may be specified in a different unit than the selected time unit by using a suffix character. The suffix character definitions are:

- l (milliseconds)
- s (seconds)
- m (minutes)
- h (hours)
- d (days)
- w (weeks)

Example
Specifying *2h* for a simulation time referencing property converts to the value *120* if the selected time unit is *Minute*.

10.3. Principal script method names

Method name	Type*
Animation	G
ArrivalAccept	S
Chart	S
Clock	G
Delay	S
DepartureTime	E
Destination	L
DestinationPosition	L
EntityCreate	L
EntityProcure	S
EntryTime	E
Evaluate	D
GetNextItem	C
Host	E
Icon	ES
Identifier	E
Index	SLHDP
LinkCount	S
Links	S
Name	SLHDP
Names	C
Origin	L

Method name	Type*
OriginPosition	L
Pause	G
Parameters	D
Priority	SH
Random	G
ResidentCount	SL
ResidentPosition	E
Residents	SL
ResidentSwap	SL
Route	E
Run	H
Signal	G
SignalSend	S
Statistics	SLAV
StatisticsReset	G
Stop	G
Terminate	H
Text	S
Time	G
Transfer	E
TravelCount	L

* G = General, S = Server, L = Link, E = Entity, D = Distribution, H = Schedule, P = Picture, A = Attribute, V = Variable, C = Class