Recommendations for Creating a GoldSim Style Guide

GoldSim Technology Group LLC

Last updated January 15, 2020

Table of Contents

Table of Contents 2
1. Introduction 4
2. Naming conventions 5
2.1 Element IDs 5
2.2 Array Label Set Names and Array Labels 9
2.3 Result Labels 10
2.4 Global Properties 10
2.5 Scenario IDs 11
2.6 Other Names 11
3. Writing Expressions 13
3.1 White Space 13
3.2 Conditions 14
3.3 Decimal Numbers 15
3.4 Simplify Expressions 16
4. Model Organization 17
4.1 Choose Appropriate Elements 17
4.2 Graphics Pane Layout 20
4.3 Using Containers 23
4.4 Input Data 24
4.5 System Model 28
4.6 Information References 29
4.7 Model Results 33
4.8 Element Fill Colors 33
5. Testing 36
5.1 Verification 36
5.2 Mass Balance 38
5.3 Calibration 39
6. Images and Graphics 40
6.1 Element Icons 40
6.2 Embedded Images 41
6.3 Influence Line Colors and Shapes 43

7. Memory Used 44
7.1 Disable Final Values and Time History Results 44
7.2 Time Step Settings 46
7.3 Delete Unused Elements 46
7.4 Delete Unused Array Label Sets 47
8. Documentation 48
8.1 Display Units 48
8.2 Text Blocks 48
8.3 Element Descriptions 49
8.4 Container Heading 49
8.5 Container Description 49
8.6 Section Borders 51
8.7 Hyperlinks 51
8.8 Notes 52
8.9 Influence Line Labels 53
8.10 Simulation Information 54
9. Dashboards 55
9.1 Colors 55
9.2 Sizing and Alignment of Dashboard Controls 55
9.3 Tooltips 57
9.4 Dashboard Hierarchy and Navigation 57
9.5 Embedded Charts 58
10. Versioning 60

1. Introduction

This guide provides ideas and recommendations for consistent conventions, organization and documentation of a GoldSim model. This guide can be used to assist with writing your own style guide or set of standards for building GoldSim models within an organization. While consistency with a style guide is important, consistency within your project is more important. Keep in mind that sometimes these recommendations may not be applicable. Use your best judgment when applying a consistent style to the organization and documentation of your model.

The intended outcome of using a style guide is to have a model that is readable. If you use styles that are different from ideas and options provided in this guide, that is fine so long as it produces the same result: a model that is easy to read by others.

This guide is organized into sections that describe various concepts that make up a model. Your model should be a document that can be read by others in an organized way, like you are telling a story about the system being modelled and provide clear steps on how to use it. It's important to consider your audience throughout the process of building and documenting your model. Typically, those readers with little modeling experience will only view the interface but other modelers on your team with more experience will need to understand the functionality. A clearly documented and well-organized model with consistent styles will provide your readers with the ability to understand what the intended functionality is.

2. Naming conventions

In a GoldSim model, you will be providing custom names for various types of objects throughout the model. The most common type of object you will work with in GoldSim are elements. Because these are referenced in expressions throughout your model, they should be easy to read. It is important to be consistent with this style (or a style of your own) for all elements in your model, especially if you are working together with others.

2.1 Element IDs

Element IDs are the most common names in a GoldSim model, which can only include letters, numbers, and the underscore ("_") character. They cannot include spaces and must begin with a letter. In addition, some names (e.g., sin, cos, min) are reserved for use as functions and cannot be used as an element name. Two types of elements (Result elements and Dashboards) have slightly more flexibility in terms of their names, which can include spaces.

There are 3 recommended ways to write element names:

- Capitalized camel case (i.e. `EvapRate`)
- Capitalized with underscores (i.e. `Evap_Rate`)
- Small case with underscores (i.e. `evap_rate`)

We recommend that you choose 1 of the 3 options shown above and stick with it throughout the model. Note: *The examples shown in this document use capitalized with underscores, but you can replace these with either of the other 2 if you prefer.*

In general, follow these rules of thumb for names of Elements in your model:

- Abbreviations are okay if obvious to the reader, such as `Evap_Rate` (meaning "evaporation rate")
- Avoid initialisms and other abbreviations that are unique to your model or difficult to decipher by the reader

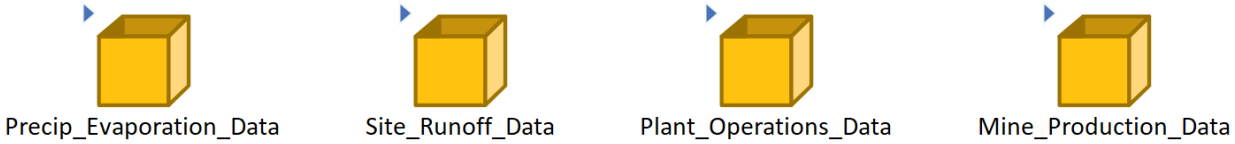
Use your natural language to tell a story in a way that you might do so when talking. In English, write names that follow a convention of attribute followed by object (`Red_Car`) or object followed by function (`Airplane_Flies`). One exception is when naming elements located inside a localized Container, where the attribute will likely follow the name of the Container that represents a physical object. Refer to the "Containers" Section for more information.

If you are using capitalized words with underscores, then you might want to avoid capitalizing some small words like prepositions in a name like "`Supply_to_Pump`". This would not be the same for camel case though: "`SupplyToPump`".

There are some additional ideas to consider for different types of elements, described below.

Global Containers

Global (open) Containers can use verbose, full names with underscores. These names are not used in expressions within the logic of the model so long names will not encumber expressions. The only constraint is the space provided in the graphics pane. In the examples shown below, we don't need to abbreviate words more than we have already.

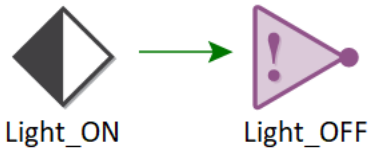


Localized Containers

Localized (closed) Containers may need to use names that are more concise since they will be used in expressions when referring to internal outputs from outside of the Container. Often, the localized Container represents a physical object or location. If there is an element called "`Population`" inside the Container called "`Seattle`", then you must refer to the output outside of the Container using the name "`Seattle.Population`". For more discussion of how to use localized Containers, refer to the Model Organization section.

Conditional Variables

Model variables with a main output of a conditional type such as Status, And, Or, Not, and an Expression with conditional output type can be appended with a word that represents its true conditional output. The word used to define the condition can be in ALL CAPS for more emphasis. For example, if a Status element is used to report the state of a light bulb, it would be: `Light_ON`



Other examples of these names include: `Pump_ON`, `Operation_FAILED`, `Gate_OPEN`, `Engine_STANDBY`, `Tank_FULL`, `Tank_EMPTY`, `High_Target_ENABLED`

Constants

A constant in GoldSim is a value or array of values that never change during all model simulations. Scenario data and data controlled via dashboard controls are not considered constants. Don't confuse a model constant with a variable that remains constant during a realization or scenario. In this section, I am referring to true constants, which would never change during all realizations and scenarios.

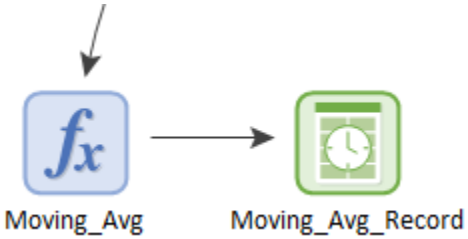
It can be helpful to write the names of true constants in all-caps with underscores separating words, such as `FRICTION_FACTOR` or `VISCOSITY`.

Lookup Tables

To help differentiate Lookup tables from other types of input data, you can append the word “Table” on the name such as `Elev_Volume_Table` or `Cost_Table`. The benefit of doing this is that it provides context within expressions, which differentiates this type of input from built-in functions. The downside to using this approach is that it makes the name longer, which complicates the expression.

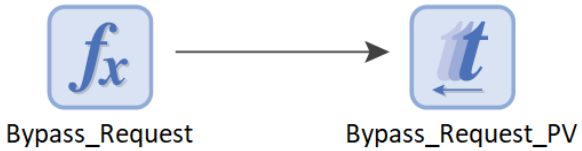
Time Series Recordings

Time Series elements provide an advanced option in which you can read the output of another element in GoldSim, and "record" the results. An easy way to differentiate a recording Time Series element from others is to append the word “Record” on the end of the element name.



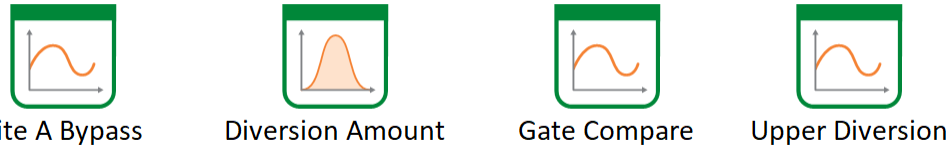
Previous Values

The input of a Previous Value element can only be defined by a link to a single output so it is helpful to give it a name similar to the output it is referencing, along with “PV” or “Previous”. For example, if the Previous Value element is referencing an output called “Bypass_Request” then you could call it `Bypass_Request_PV` or `Previous_Bypass_Request`.

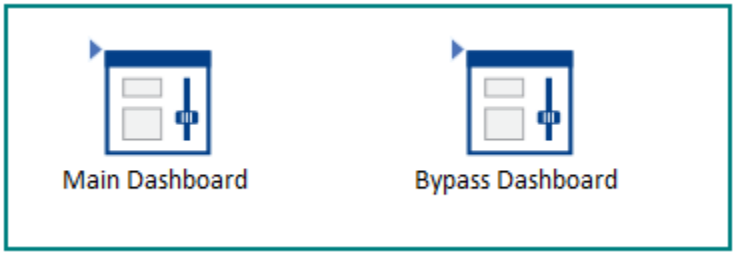


Results, Dashboards, and Hyperlinks

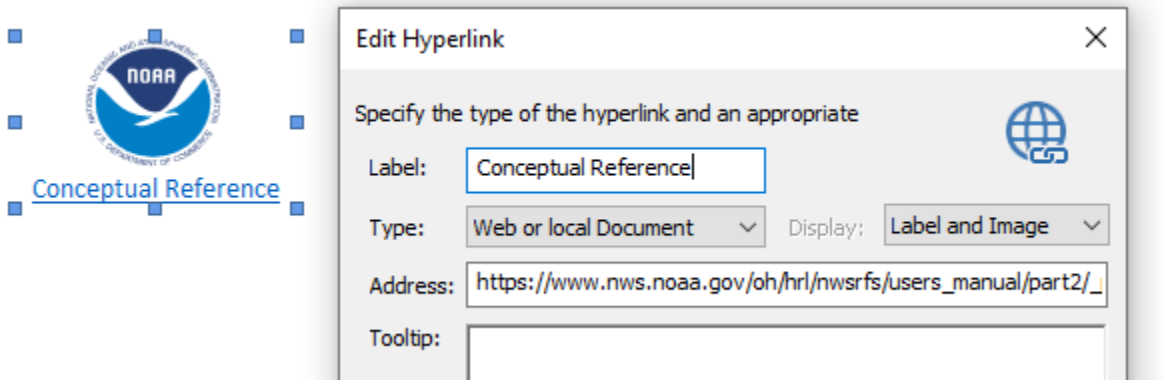
The names of Result and Dashboard elements allow for spaces in the names. Because of this, you should use spaces when more than a single word is used. By default, the name of the element is displayed in the header of the chart so you should make the names as descriptive as possible within reason.



Append the word "Dashboard" to all dashboards.



It is recommended that Hyperlinks in GoldSim also follow these same conventions of Capitalized words separated by spaces.

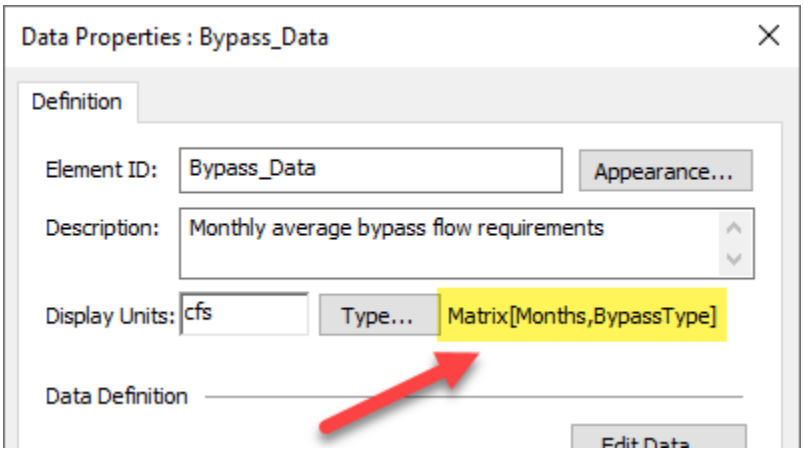


Differentiate Between Types of Information

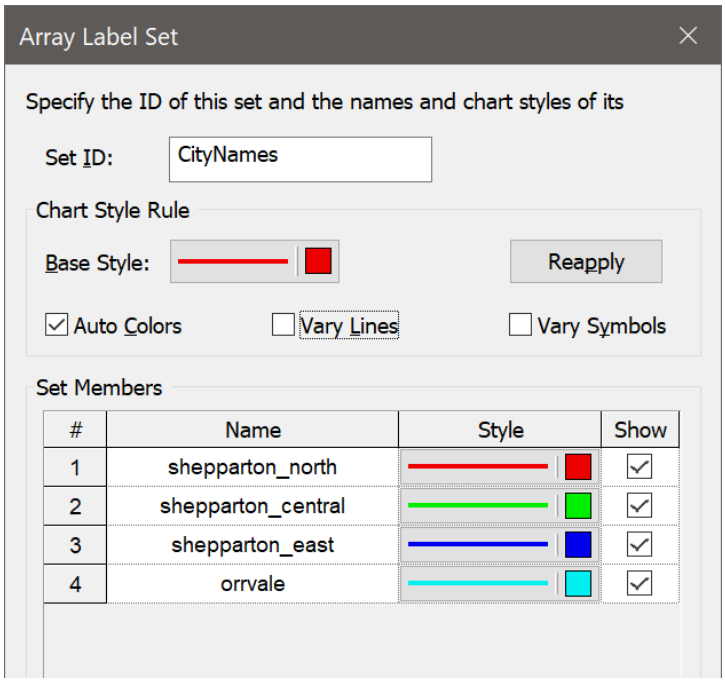
In some cases, it might be helpful to differentiate between types of information in your model. For example, you might have elements that refer to concentrations (or flows) and you want all elements of this type to have a similar name. To do so, for example, you could add "_C" or "Conc" (and "Q" or "Flow") to the beginning or end of these element's names.

2.2 Array Label Set Names and Array Labels

It is helpful to create concise names for Array Label Sets because they appear in the Type display for non-scalar elements. Using CamelCase and abbreviations are good ways to shorten the names.

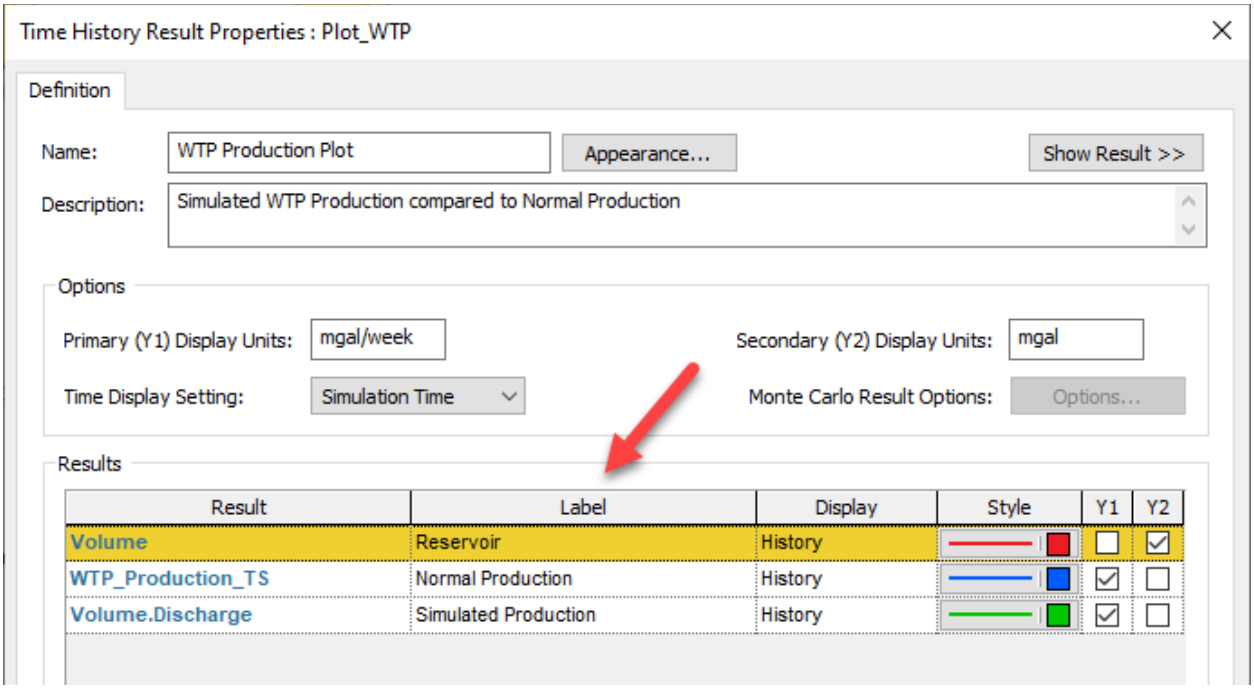


To reduce confusion between element IDs in your model and array item names in arrays, the names of array labels should follow a different convention than element IDs. In the example below, lower case words with underscores are used for the array item names.



2.3 Result Labels

This is for labels of outputs connected to Result elements. You can modify the Label for each result (which can subsequently be displayed in legends and headers, footers and axes labels). Use capitalized words with spaces.



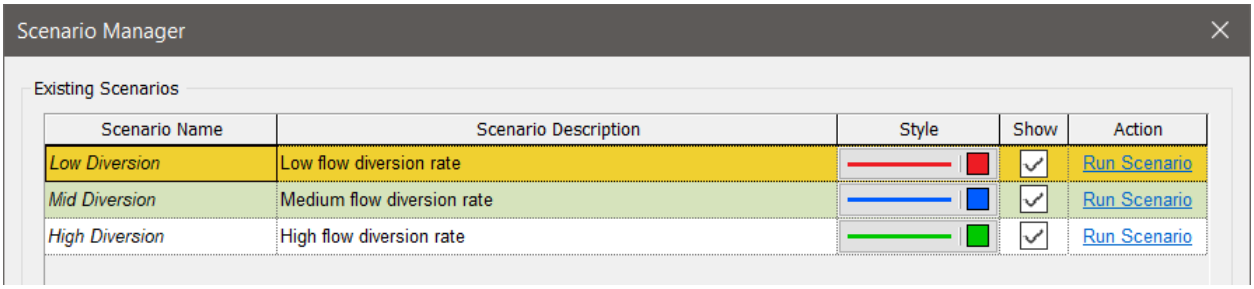
2.4 Global Properties

In some cases, you may want to define global properties that can be referenced throughout your model. Of course, one way to do this is to define Data elements at the global level of the model (i.e., outside of any localized Containers). GoldSim also provides a more convenient way to do this via the Globals tab of the Simulation Settings dialog. Names of a global property have the same restrictions as an element ID.

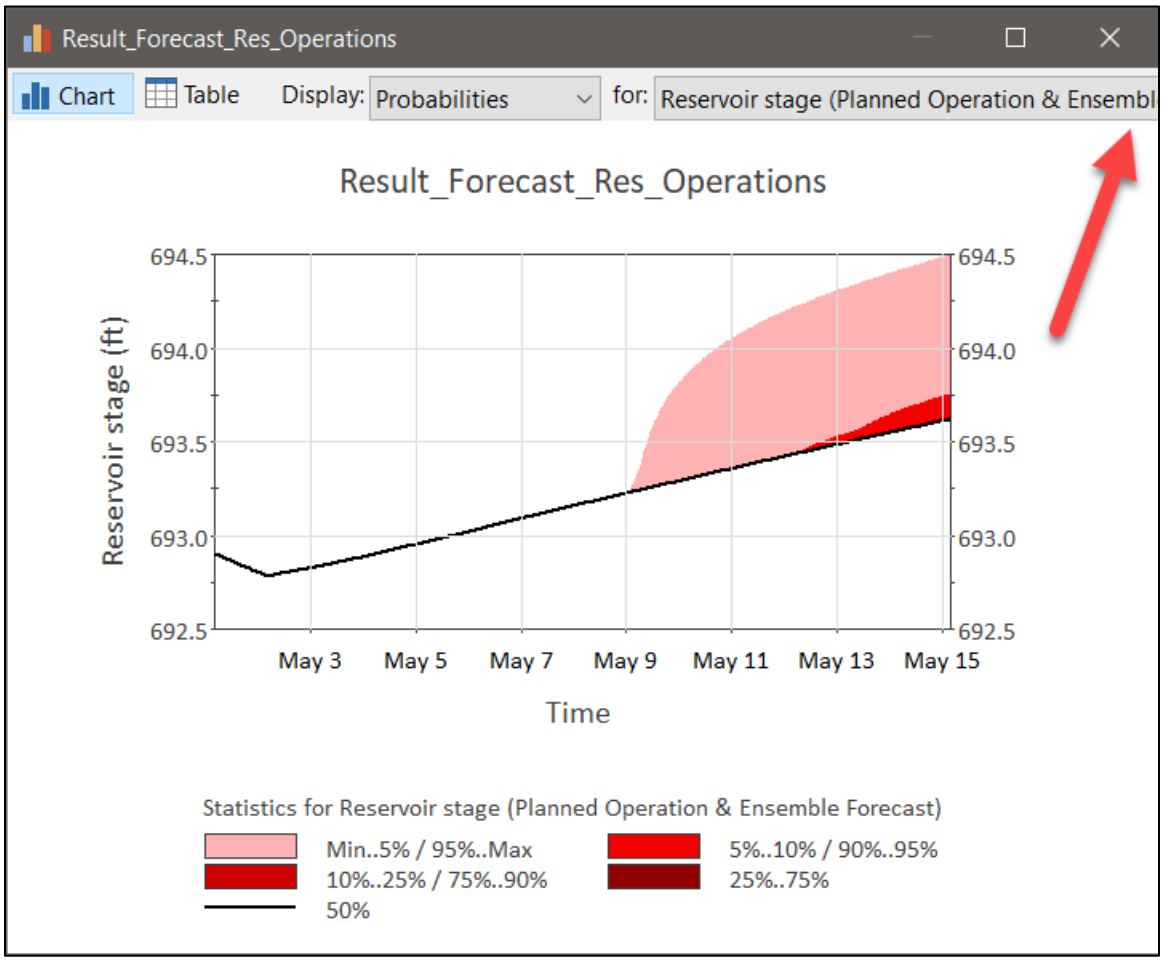
A good way to differentiate the names of global properties from others is to use ALL_CAPS separated by underscores.

2.5 Scenario IDs

Scenario ID names should be concise because they appear in various UI controls, the status bar, results and Dashboards. It is recommended that you use Capitalized words separated by spaces as shown in the example below.



If you use names that are too long, you will find some controls must stretch to show the full length and cannot fit well (as seen in the example below).



2.6 Other Names

There are a few other places where you can specify user-defined names in GoldSim, as shown below. For these, use the same conventions as other element names.

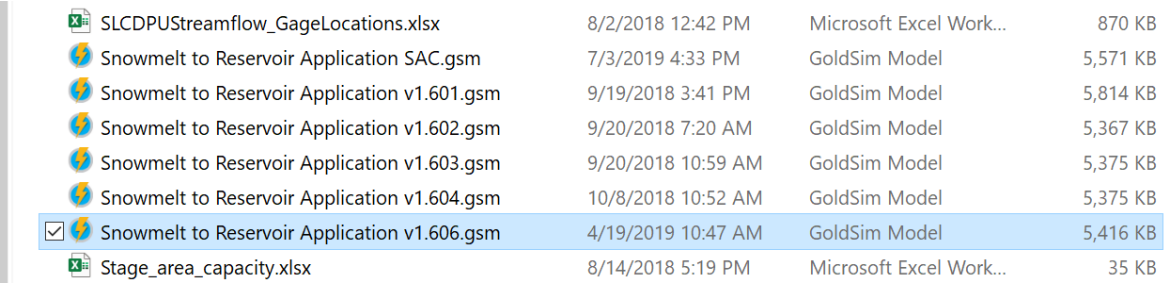
Custom Units of Measurement

One of the more powerful features of GoldSim is that it is dimensionally-aware. GoldSim has an extensive internal database of units (all of these units are listed in **Appendix D of the GoldSim User's Guide**). You can also create your own units. Any unit that you create on your computer automatically becomes a system unit (i.e., it is stored in the system units file, and hence is subsequently available to any other model you open on your computer).

When creating a new custom unit, make sure you are using the correct case and spelling that is typically found in literature and published standards.

File Name

The name of the GoldSim model file is important because you will likely be creating copies to share the file and need to avoid name conflicts. Capitalized words with spaces or CamelCase are good ways to name the file, followed by an identifier for a specific update to the file. Many people prefer a date, but another option is a version number, as shown below.



Exposed Output Alias on a Localized Container

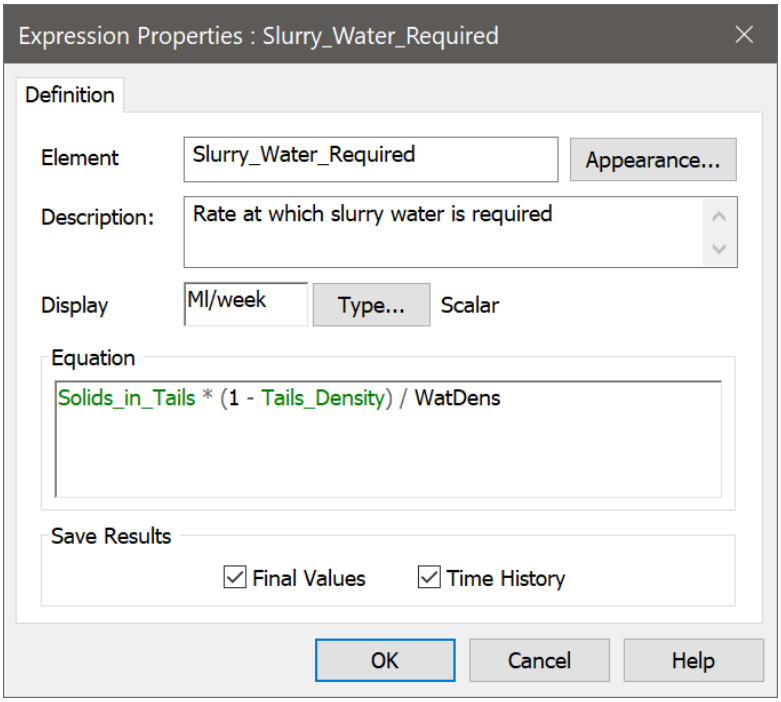
Try to avoid custom alias names on localized Containers as this can lead to confusion if element names inside the container are changed later on. However, in some cases (when multiple exposed outputs have the same names) this will not be possible, and custom alias names will need to be used.

Submodel Output Interface Labels

Use a similar convention as described for exposed outputs of localized Containers.

3. Writing Expressions

GoldSim provides several features that enable you to enter mathematical expressions into input fields. Become familiar with and make use of these features to write expressions efficiently. This section introduces our style recommendations for expressions written in GoldSim.



3.1 White Space

The way you use white space in your expressions can make a significant difference in readability. Consistent and careful use of white space can significantly increase readability by making math operators more visible, improving text wrapping, and creating emphasis on important parts of the equation. As shown in the ensuing examples, expressions that have no spaces and are more difficult to read.

The general rules of thumb for using white space are:

- Before and after math/logic operators except power ("length^2")
- After a comma

Below I have listed some various use cases that compare different ways of using white space with the recommended way indicated next to the "Good" label.

Math Expressions

Bad: `Width+Height*(1+X_Left^2)^0.5+y*(1-X_Right^2)^0.5` ✗

Good: `Width + Height * (1 + X_Left^2)^0.5 + y * (1 - X_Right^2)^0.5` ✓

(note, this is not a real equation)

IF Statements

Bad: `if(Fraction_of_Treated*Inflow>=Inflow,Pump1,0l/s)` ✗

Good: `if(Fraction_of_Treated * Inflow >= Inflow, Pump1, 0 l/s)` ✓

Vector and Matrix Expressions

Bad: `vector(ore[1],waste[3,2])` ✗

Good: `vector(ore[1], waste[3, 2])` ✓

Values with Units

Add a space before a unit that follows a value.

Bad: `max(0ft, Max_Pumping - Current_Pumping)` ✗

Good: `max(0 ft, Max_Pumping - Current_Pumping)` ✓

3.2 Conditions

GoldSim allows you to alternatively use words "then" and "else" in IF statements if you want. If you use these words, then obviously, you would need spaces:

```
IF(Month == 4 then Outflow else 0.0 l/s)
```

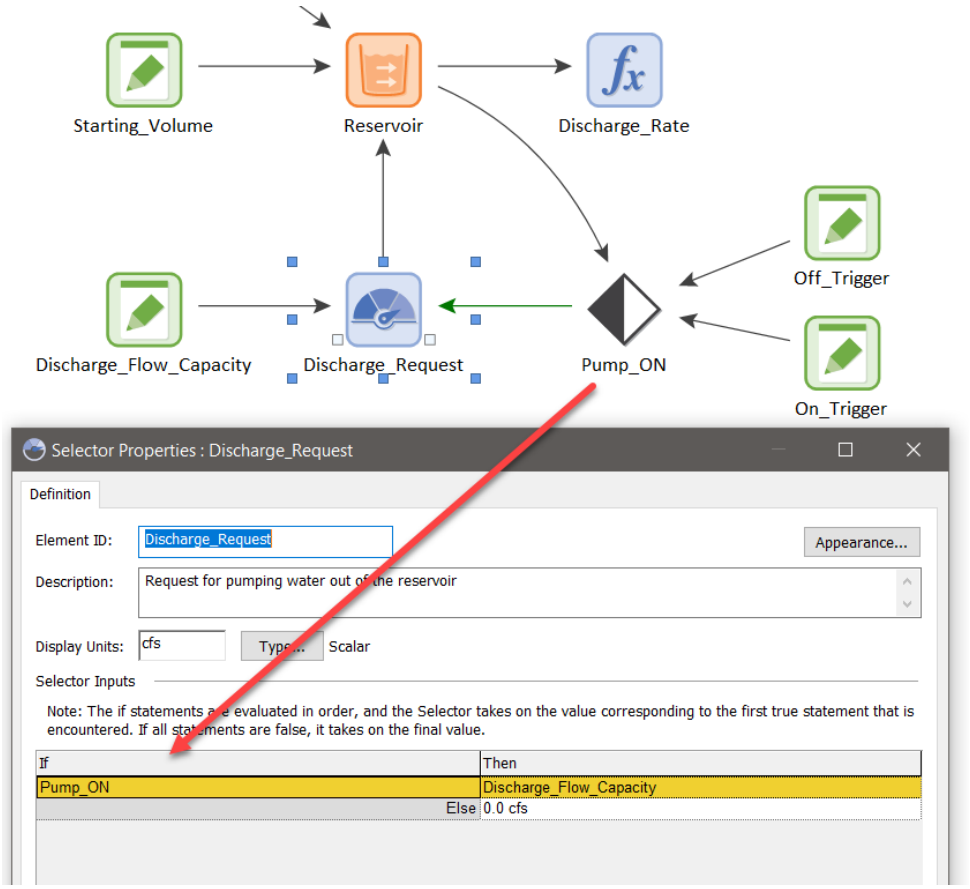
✓

If you choose to do this, you should be consistent (either use it for all IF statements or none). GoldSim uses 2 different data types in expressions: value and condition. You can use a condition type output to express a condition in IF statements. Do not duplicate the condition by checking for equality to True or False. For example, the below equation can be written in 2 different ways:

Bad: `IF(Pump_ON == True, Outflow, 0.0 l/s)` ✗

Good: `IF(Pump_ON, Outflow, 0.0 l/s)` ✓

Because the first method is redundant, it is better to write it with only the conditional output.



3.3 Decimal Numbers

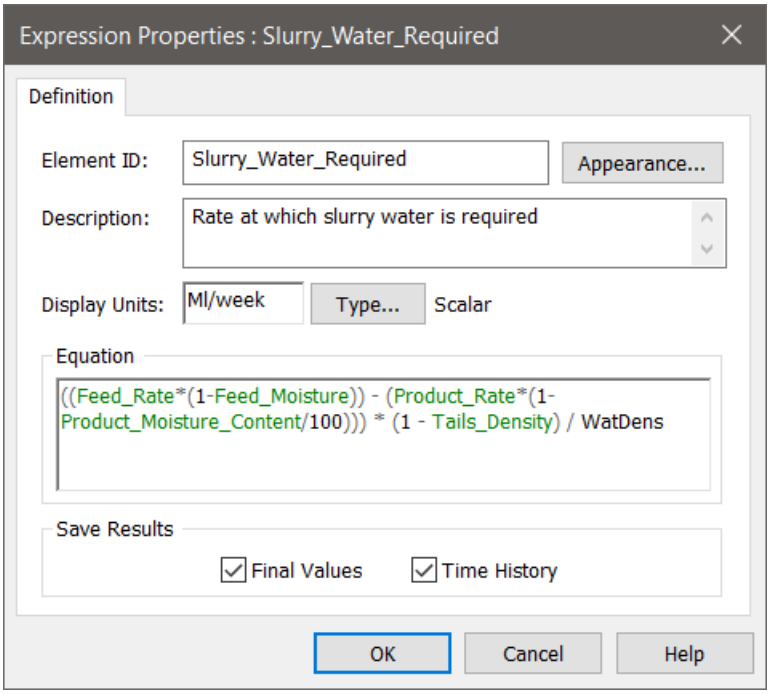
For whole numbers, either show the integer or a trailing zero after the period. Do not just include the period without a zero because this is harder to read.

Bad: `Flow_Rate = 3. l/s` ✗

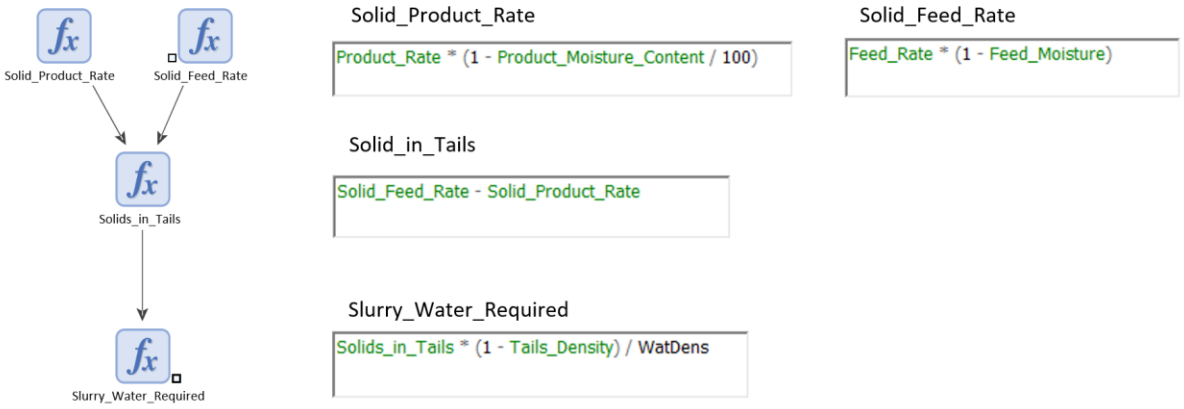
Good: `Flow_Rate = 3.0 l/s` ✓

3.4 Simplify Expressions

Don't hide the details of your model within long input expressions. It is better to add a few additional elements such that the relationships in the model can be shown graphically and are transparent. It can be difficult to read and understand long equations with layers of sub-expressions buried in parentheses. If you find yourself trying to decipher where your closing parenthesis is, there is a good chance you should consider breaking up the expression into smaller parts. Take the expression below, for example. This expression isn't terribly long, but it is difficult to make sense of it because of so many embedded expressions within parentheses.



Instead, this expression can be broken out into the following elements:



4. Model Organization

Model organization is one of the most important aspects of your model. Without it, your readers will be lost, unable to make sense of how the model is set up. This can lead to many extra hours of time in debugging and making changes to the model. Because it is so easy to add elements to a model, if you don't spend time organizing the model, you could end up with a large space of jumbled elements thrown across the graphics pane.

When you begin to design your model, you should always have in mind the audiences to whom you will be presenting or explaining the model. Most models will have at least two audiences:

- a high-level audience (managers and decision-makers who are primarily interested in the "big picture" or "bottom line"); and
- a low-level audience (analysts who are also interested in the technical details and assumptions of the model).

Often, there will be multiple low-level audiences, which are interested in different technical aspects of the model. There may also be a "mid-level" audience, which is interested in more than the "big picture" but does not want to get lost in the details. For a model to be successful (i.e., useful), you must be able to present and explain your model effectively to all these audiences.

One way to organize the model is as shown below. You can refer to this organization and modify it to fit your model's requirements. This section will walk through some general guidelines for each of these categories.

- Dashboards
- Input Data
- System Model
    - Use 1 localized container for each system component
- System Variables
- Results
- Testing

GoldSim provides the tools for you to create a single model that can be explained and presented to multiple audiences. The primary way in which you accomplish this is by organizing your model into a logical, top-down hierarchy.

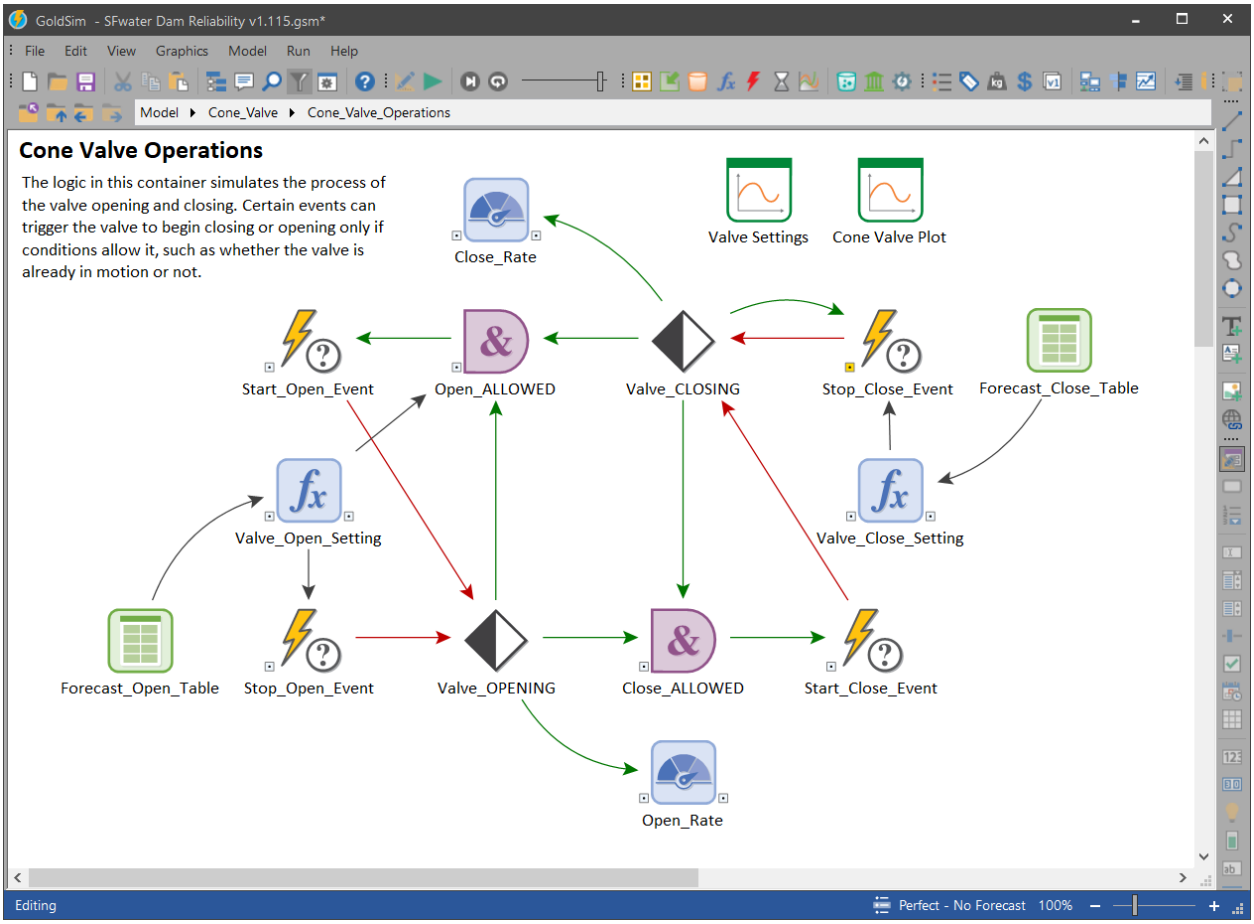
By following the guidelines in this document and sticking with them as you continue building and maintaining your model, you should avoid problems and it will pay off in the long run.

4.1 Choose Appropriate Elements

Use elements that fit the context of what you are trying to accomplish. GoldSim provides a large set of various elements for a reason. While it might be possible to build most of your model's logic simply using Expression elements, you should avoid doing this because the different types of elements provide visual context for the reader. For example, it is possible to use Expression elements for defining input data, IF statements, conditions and sums. You should instead use Data elements for data, Selectors for

IF statements, And/Or/Not elements for conditions and Sum elements for sums, where possible. Below are examples showing what not to do compared to the recommended approach.

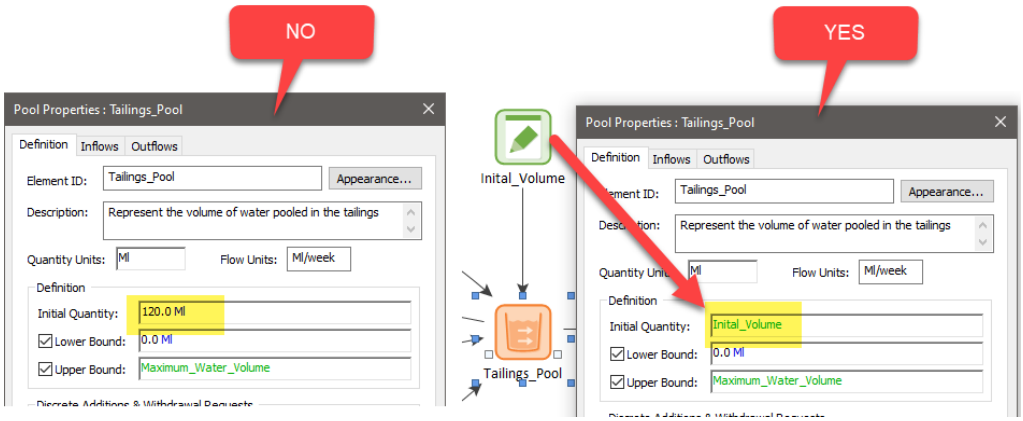
An example of using different GoldSim elements is shown below. In this example, simple Expression elements could have been used in the place of many of what you see. The specific GoldSim elements that are used below provide some insight into how the model works without even drilling into the specific expressions for each of these.



Try to avoid using the Expression element too much when other elements can do it much better. Below, I provide some examples of what elements are more appropriate than others. In general, you should learn about all the GoldSim elements and use your best judgment which ones are best for each occasion.

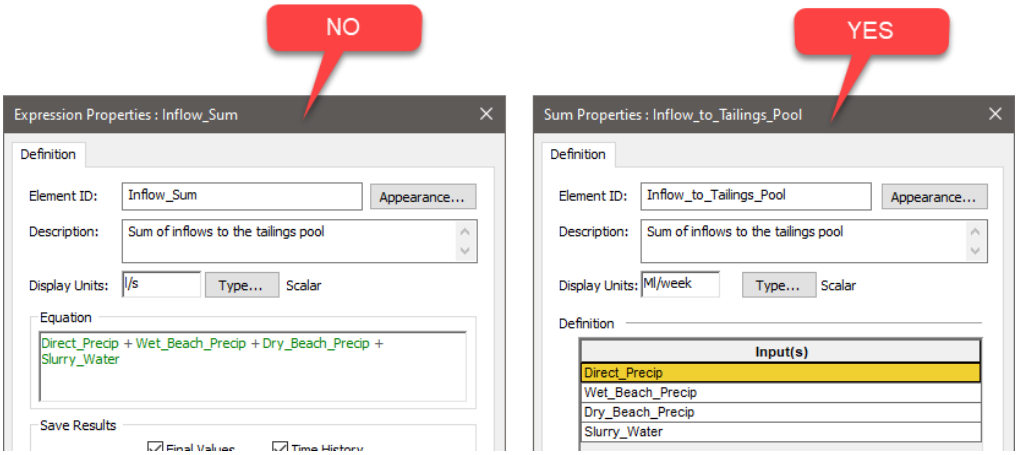
Data Element

Avoid inserting numeric values inside elements that are not Data elements. Use a Data element to hold the value then refer to it in your other elements as shown below.



Sum Element

If you are summing outputs from multiple elements, choose the Sum instead of the Expression.



Material Delay Element

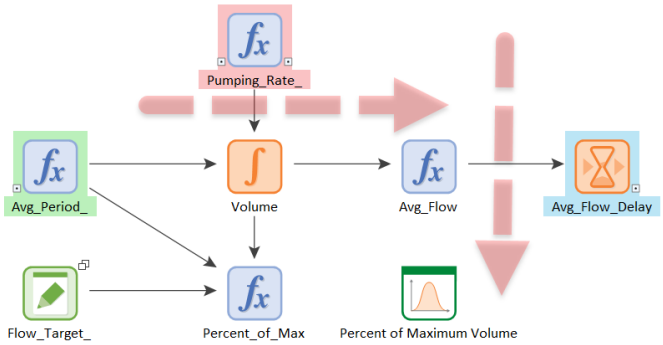
Material Delay elements are intended to be used to simulate delays in the physical movement (flow) of material. These delays often have a critical impact in the dynamic behavior of systems. Do not use an Information Delay or Previous Value element in its place if you intend to delay material flows.

4.2 Graphics Pane Layout

This section describes ideas for the style of the graphics pane, keeping in mind that it should be consistent throughout the model.

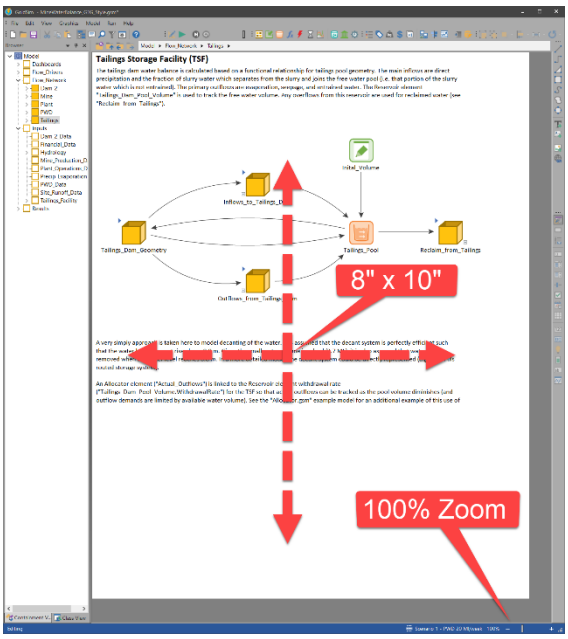
Flow of Logic

As a rule of thumb, try to organize elements within the graphics pane so that the reader can walk through the logic in a left to right and top to bottom direction. Below is an example of elements that follow this pattern of direction.

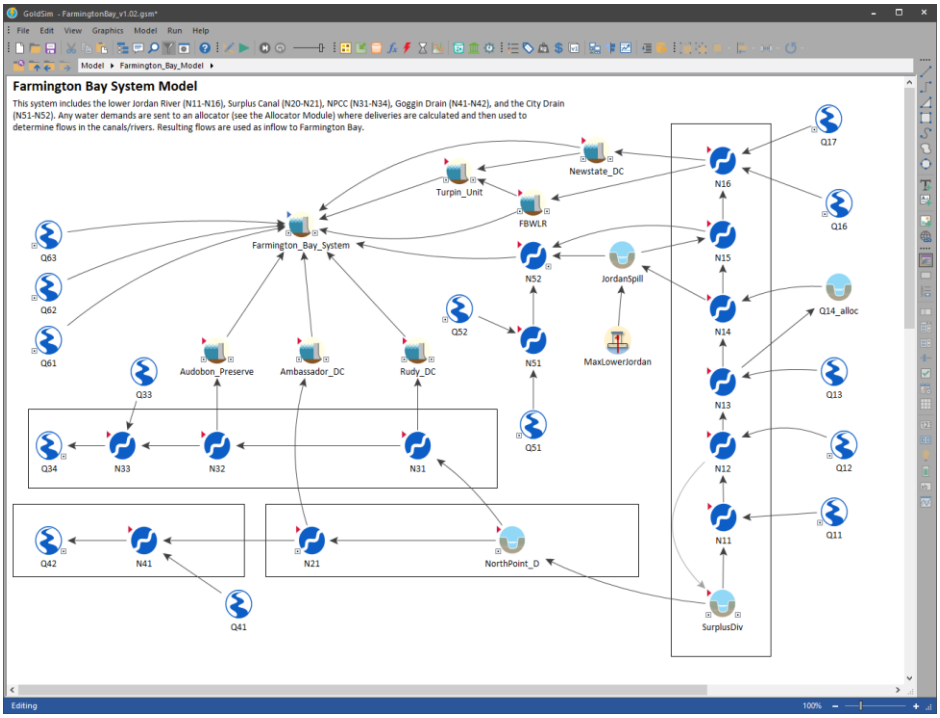


Dimensions

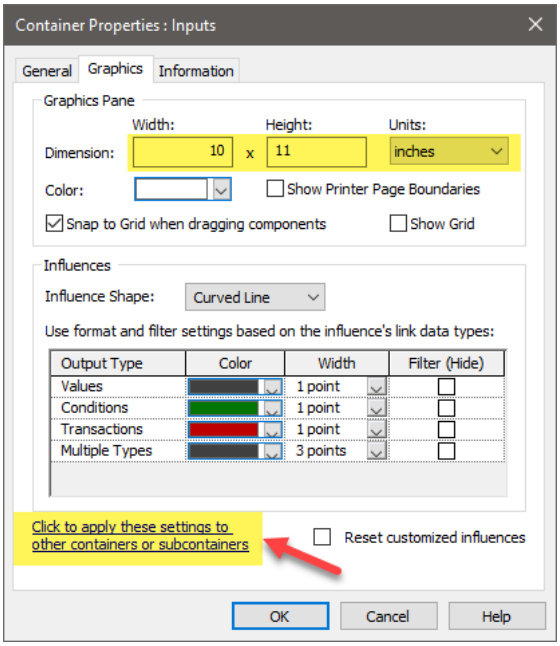
Try to fit everything within a consistent space for most containers of the model. One option is to fit everything in the container on a standard 8.5 x 11 size paper at 100% zoom in the graphics pane. A good rule of thumb is that you try to keep the total number of visible elements to **20 or less**.



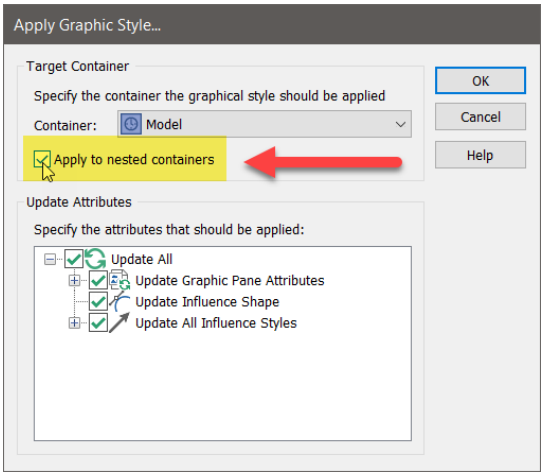
An exception can be made for a container showing a representation of a large system, especially when overlaid on a map image that requires the model elements appear smaller. In the example shown below, a larger area within the graphics pane is used to show all the nodes of a water system. Exceptions like these should be allowed to provide a schematic of the system being modelled. For most models, there will only be one Container with this exception.



If you want, you can explicitly define the bounds of the graphics pane with width and height values and apply these to all Containers. You can automatically apply these settings to all Containers by clicking the link at the bottom of the Graphics tab of the Container properties:

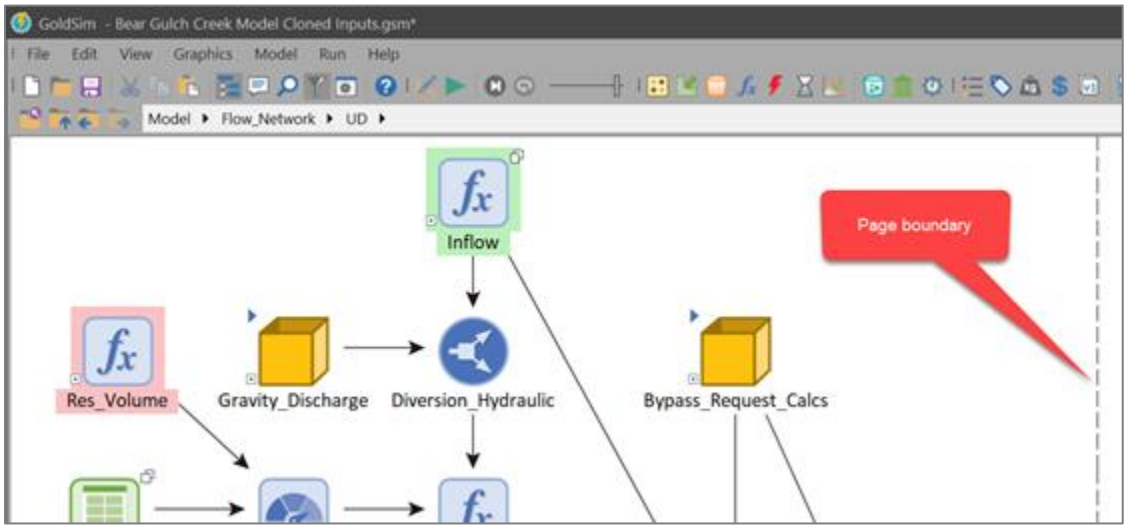


After clicking on this link, make sure you check the box to apply to nested (child) Containers.



This is one way to enforce that all elements are confined within a limited area that doesn't require the reader to zoom out or pan around to see what is going on. If you find this method to be too restrictive for some Containers, you should make some exceptions to the rule but overall, this restriction will make the model more consistent and easier for everyone to read. It will also force you to divide the model up into smaller components that are easier to understand.

Alternatively, you can use Printer Page Boundaries instead of pane dimensions to guide you on the width to use. Below is a screen capture of an example model, showing how the documentation and elements fit within this default page width. Limit number of elements shown in a single Container (avoid requiring user to zoom out or scroll to see all elements).



Graphics Pane Orientation

The orientation of the working space of the graphics pane depends on how people will most likely read your model. If they do it with the model open on ½ of a computer screen (either right or left half), then a portrait orientation should work best. Otherwise, a landscape orientation is best.

Graphics Pane Zoom

Zoom should generally be 100% for the entire model.

It may be tempting to reduce the zoom level on the graphics pane to accommodating a growing number of elements being added due to more complex logic. But this approach will lead to a model that is inconsistent and more difficult to read. As the complexity grows, you should use GoldSim's Containers to further encapsulate parts of the model. This way, people reading the model can view a single page without becoming overwhelmed.

Background Color

The background color of the graphics pane should be white. This provides the best contrast for other colors and images used.

4.3 Using Containers

Perhaps the most important requirement of a well-documented and easily understood model is that it be well-organized. Containers provide the mechanism in GoldSim by which you can create well-organized models. In most cases, you do this by placing model elements in a "top-down" containment hierarchy in which the level of detail increases as you "push down" into the hierarchy.

In a well-organized model, you can imagine that each Container has a specific "message" and a corresponding audience at which it is targeted. For example, a model's highest-level Container might have the following message: "This is the problem I am trying to solve, and here is the overall structure of the model I have built to solve it". Such a Container would be intended for all audiences.

Another Container's message in the model might be "Here are the detailed assumptions regarding process X". This Container's audience would be the analysts or technical personnel interested in that aspect of the model.

You "hide" these details in a Container, because they are of no interest to higher level audiences (e.g., managers). Those who are interested, however, can be shown the contents of the Container.

Localized Containers

In most models, element names are forced to be unique. In some cases, however, you may want to have more than one element with the same name. This requirement usually arises when you have several similar subsystems in your model.

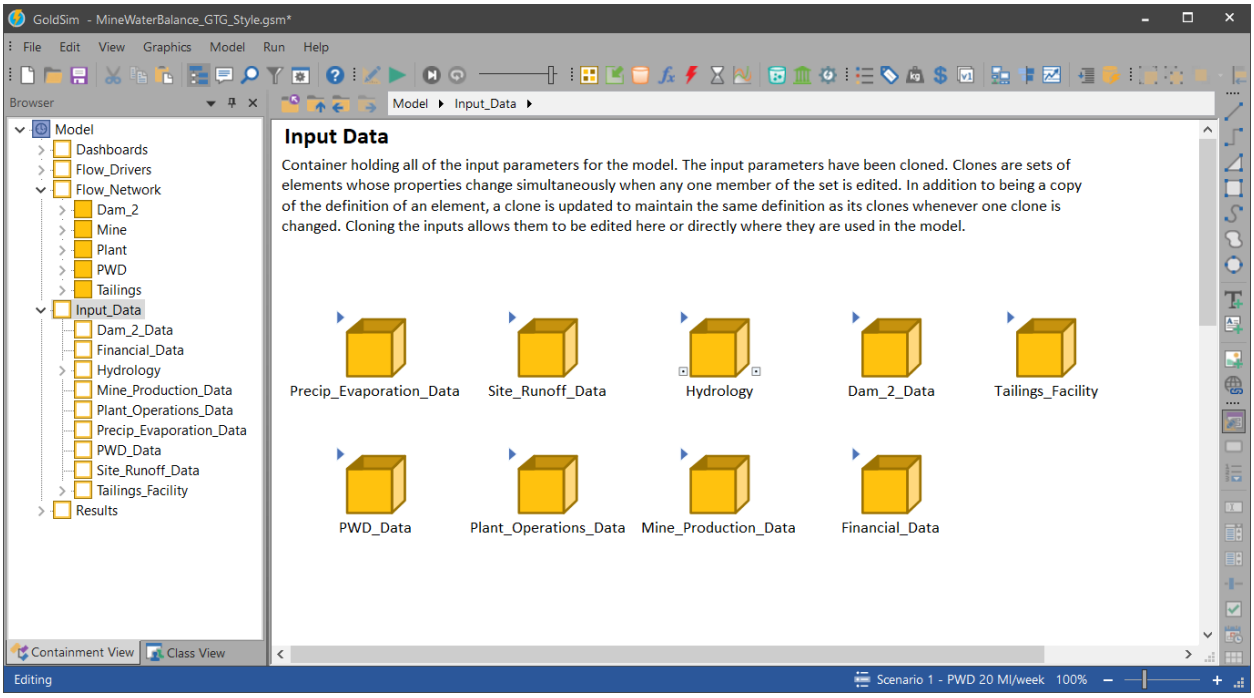
For example, suppose that you have developed a subsystem (within a Container) that calculates the balance in a bank account as a function of the previous balance, interest rate, deposits and withdrawals. Each of these items would be represented by its own element in the Container. After creating these elements, you decide you would like to track ten other accounts.

In such a case, it would be convenient to create ten copies of the Container (one for each of the other ten accounts). If GoldSim allowed you to just make copies of the bank account Container, however, there would be confusion when you referred to an element by name (e.g., in an expression), since GoldSim would have no way of knowing which copy of the element you were referring to.

GoldSim's solution to this problem is to allow you to localize portions of your model, such that any element names in the local region are hidden from elements outside of that region. A local region is referred to as a "scope" in GoldSim. Elements with the same name cannot exist within the same scope. Elements can, however, have the same name if they are in different scopes. You change the scope of an element by localizing the Container in which it is located. Unless you specifically "expose" an element, the contents of a local Container can only be "seen" and hence referenced by other elements inside the Container.

4.4 Input Data

It is recommend that all the input data in the model should be either be organized into a single folder at the top level of the model (called, for example, "Input_Data") or within the components of the model where they are being used. Below is a screen capture of a model with the input data being saved at the top level in the "Input_Data" container.



Note that input data may consist of Data elements, Stochastics, Lookup Tables and Time Series.

For models with fewer inputs, you might consider cloning the inputs to both the top-level Input_Data folder ***and*** the components of the system. Refer to the "Cloning Input Data" sub-section for more information on how to do this.

Globals

In some cases, you may want to define global properties that can be referenced throughout your model. One way to do this is to simply define Data elements at the global level of the model (i.e., outside of any localized Containers). GoldSim also provides a more convenient way to do this via the Globals tab of the Simulation Settings dialog. This is a good way to define model constants because these cannot reference other outputs of the model.

Once you have added a Global property, you can subsequently reference the variable by using the specified Name that you defined previously, preceded by a ~. For example, if you create a Global called X, you could reference it in input fields anywhere within the model as ~X.

Organizing Input Data

There are 2 ways to organize the input data at the global level:

- Organized by subject
- Organized by type of data

Organizing by subject allows the reader to browse based on the components and subjects of the model while the latter will provide a way to view the types separately. The latter approach is preferred for large models because it is often helpful to group elements that read data from external files in a similar way. For example, it might be helpful to import time series data into your Time Series elements. It would be nice to have these all in one place in case you need to make a change to the import process.

If you decided to clone the inputs into the individual containers for the system, then it would make sense to organize the inputs by data type since you can always browse through the system model and find their clone. Some examples of input data categories that might be useful (and are suggested) include:

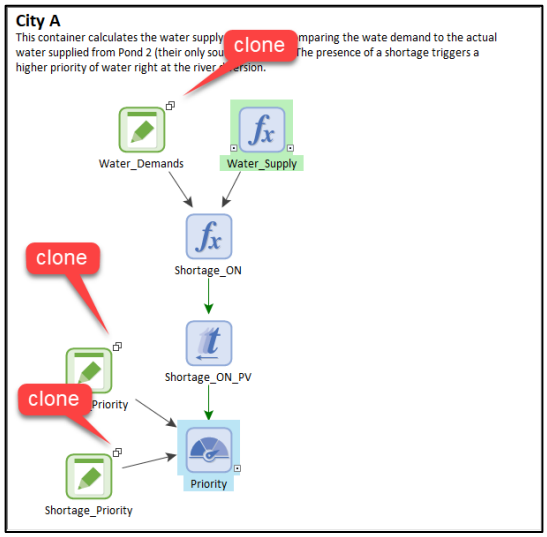
- Constants
- Key_Assumptions
- Optimization_Variables
- Scenario_Data
- Time Series
- Stochastic_Input
- Test_Data

This list is not exhaustive but intended to provide you with ideas for your model.

Cloning Input Data

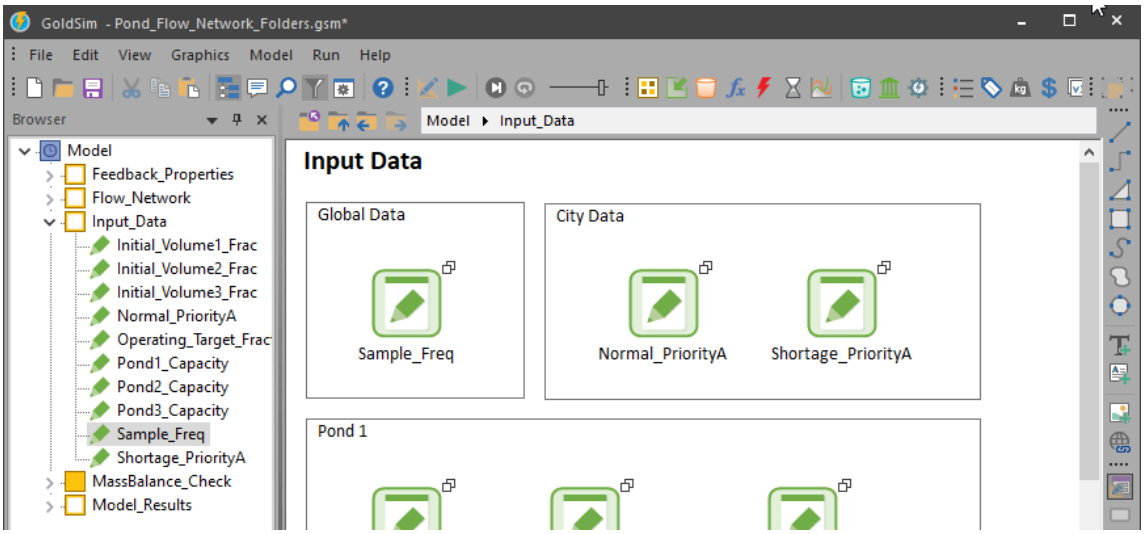
Cloning Data elements (and other inputs such as Stochastics, Lookup Tables and Time Series) to other places in the model helps the reader understand how they are being used without having to navigate back and forth between the system model and the inputs Container. If you don't have a lot of input data in your model, then this approach might be appropriate (since a cloned element requires the same amount of memory as a copy of an element).

You create clones of an element by right-clicking on the element you wish to clone in the graphics pane or a browser to access its context menu, and selecting "Clone Element". When you do so, you will then be presented with a dialog containing all the Containers in your model. Using this dialog, you must then select the Container into which you wish to place the clone. If you select a Container in a different scope, the cloned element will have the same name as the element being cloned. All clones are "equal". That is, the original element is no different than the clone which was created from it. Both are clones, and if you change an input to one, the same input in the other is automatically changed accordingly.

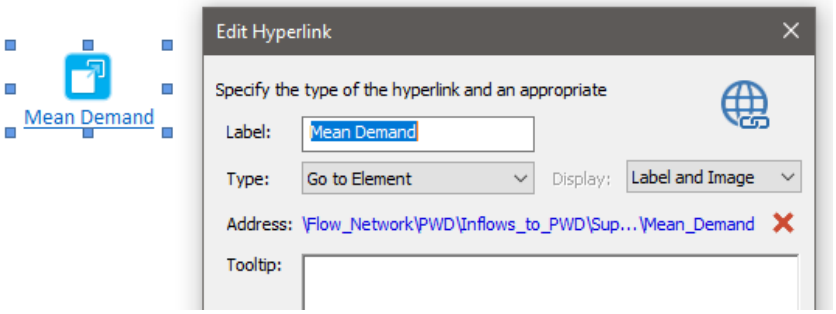


The benefit of organizing the model this way is that we can work with all the data in multiple places.

Below is a view of the input data all Contained in the Input_Data Container. Note again the symbols indicating clones. In most large models, the input data needs to be further organized into child Containers. Localizing these child Containers is recommended.

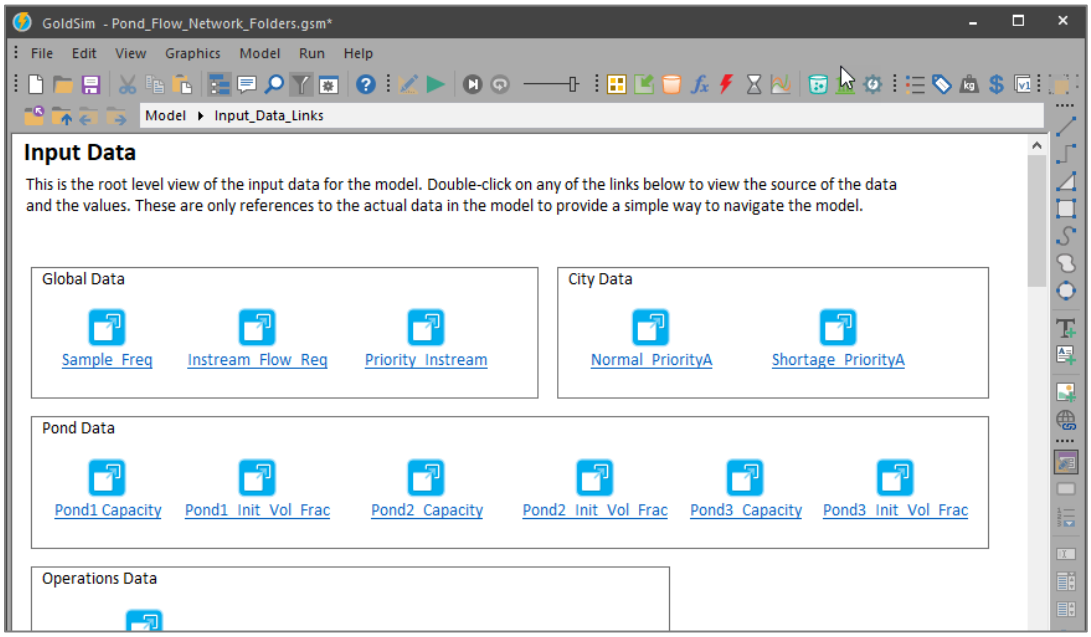


Note: Scenario data cannot be cloned so you will need to use a Hyperlink in its place. Set the Type to "Go to Element" so that when you double-click on the Hyperlink, you jump directly to the scenario data.



Hyperlinks to Input Data

An alternative to cloning all input data is to use Hyperlinks that jump to the data from the top level. This is applicable to larger models that use a large amount of input data since a cloned element requires the same amount of memory as a copy of an element.

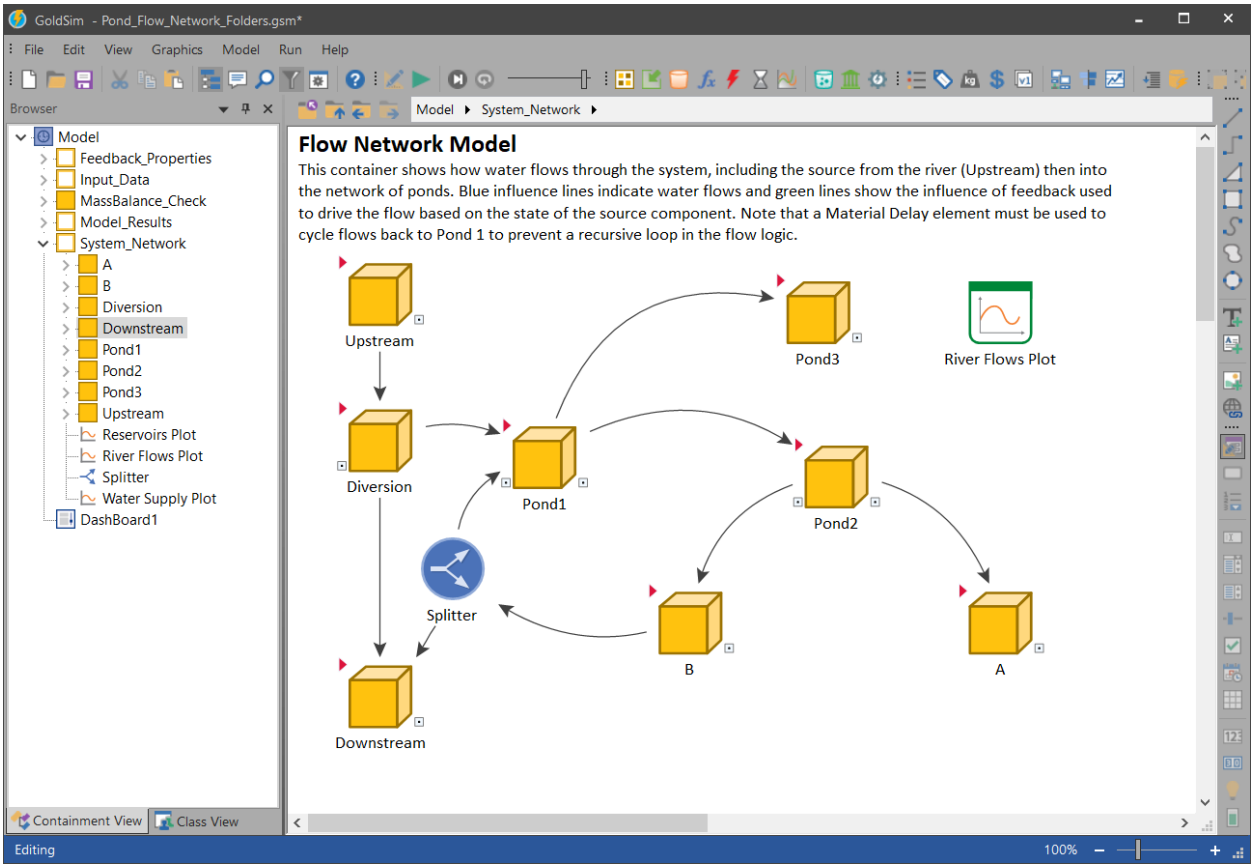


4.5 System Model

The purpose of building a model is to represent a physical system that changes over time. Most often, this system will take the shape of a network of interconnected components. It is often useful to organize the model in such a way that provides a visual representation of this system. You can do this by building components of the system inside a parent container that represents the system you are modeling. This parent Container is the “System” Container of your model.

Even though the name “System” is used in this style guide, the actual name used is not important. Choose a name that works best for your model. Each of the nodes within the system can be made up of a single element or (most often) a localized Container. If you use Containers to represent system components, they should be localized so that you can take advantage of their object-oriented nature.

Below is a screen capture of the contents of an example System Container, showing the child Containers inside that represent the components of the system and their physical interactions.

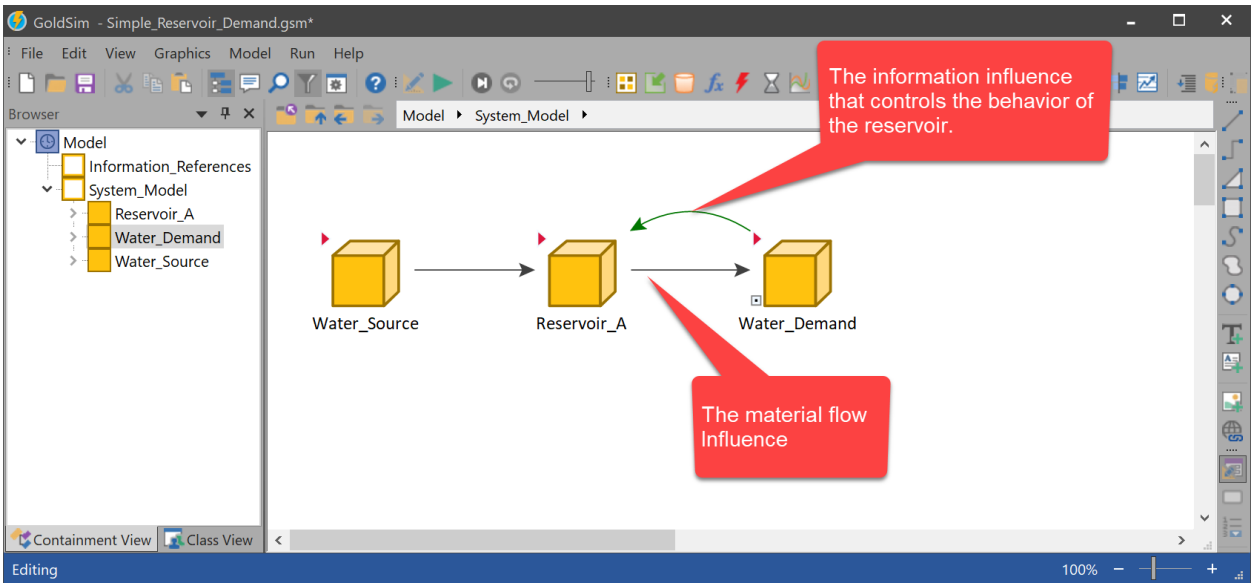


Within each node (Container) of the system, you can reference input data, variables within the node, and outputs from other sibling nodes in the system.

4.6 Information References

Sometimes attributes of a system component are used in another component to influence behavior. This kind of influence is referred to as information. It is very common in a system model to have many information-type influences and it will quickly make it harder to understand how material flows are represented by just looking at the influence lines. If you want to improve the visual representation of material flows, then you should re-route the influences to pass through a parent Container (outside the System Model Container). If you feel this type of influence line filtering is unnecessary for your model, feel free to skip it.

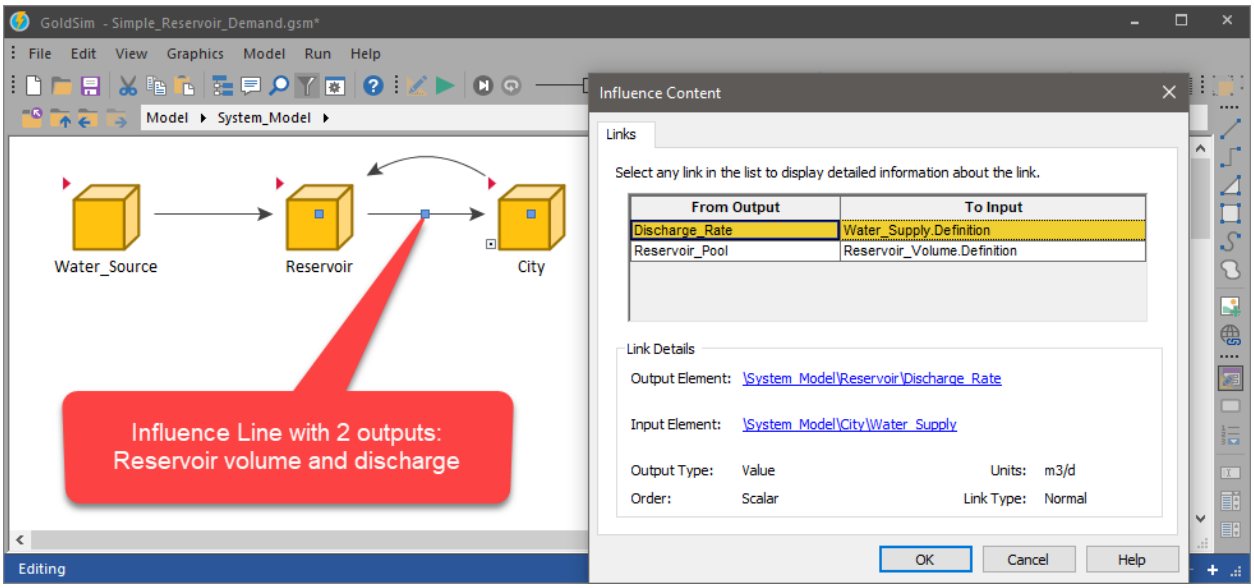
To give you a better understanding of the difference between material flows and information influences, consider the simple example of a supply-demand model, shown in the screen capture below. In this model, there are 2 influence lines connecting the Containers “Reservoir_A” and “Water_Demand”. One of the influences represents the material flow from the reservoir to the water demand. The other influence line doesn’t represent water flowing but instead the information used to control the flow from the reservoir.



It can be difficult to visualize the flow paths at the system level (of complex models) because of additional influence lines that represent model logic. To create a clearer visual representation, it is recommended that you use parent-level references to filter out the information type influence lines. This is done by referring to information outputs in a parent Container (located outside the System Container) then refer to these external references in expressions within the system. This approach is better than manually changing the colors of the influence lines because this model structure improves the overall organization of the model as well. The goal is to provide the reader with a clear view of the material flows between system components. Keep in mind that material flows are not limited to water. Your model could be simulating flows of solids, money, people, or items.

Identifying Information Influences

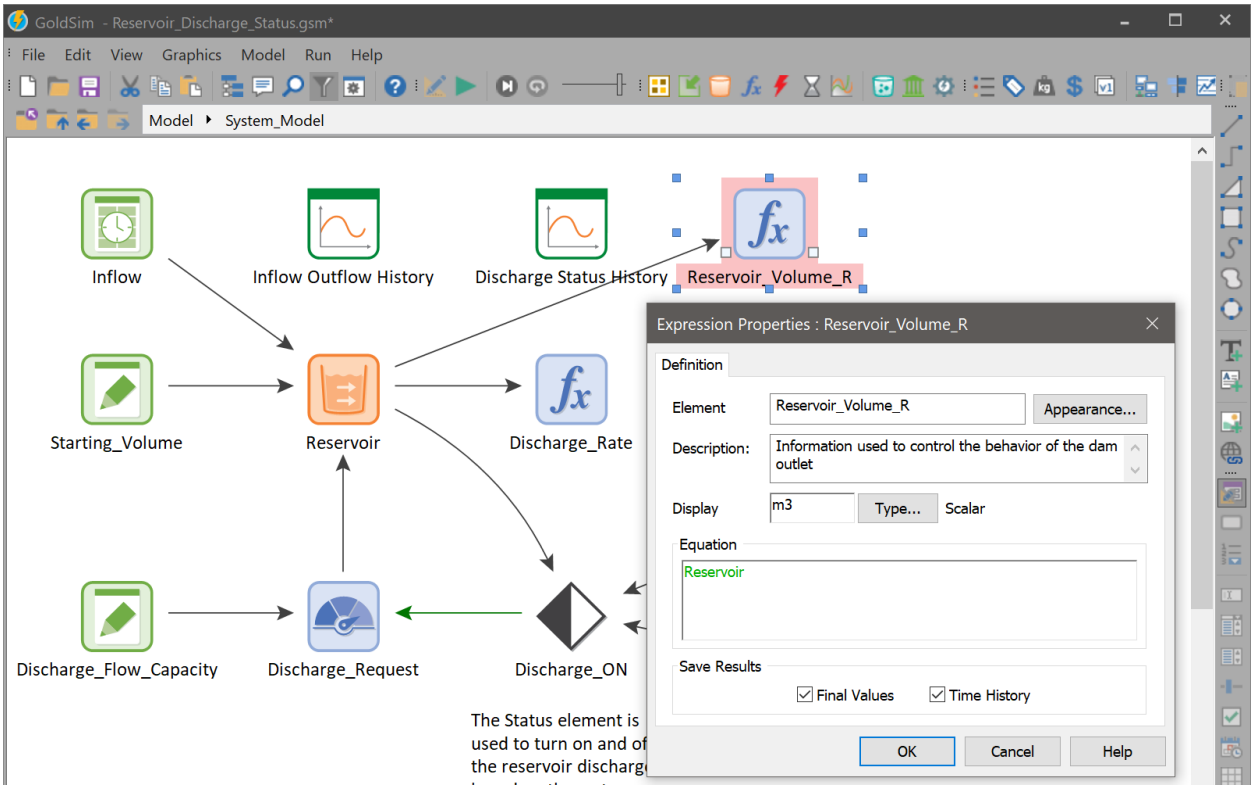
The first step in organizing a model with a focus on material flows is to identify the influences that are information only. If you double-click on an influence line, a list of outputs will be displayed, showing you all outputs that are being referenced. In the case of the example model shown below, it shows the discharge rate and volume of the reservoir that is being referenced inside the City component.



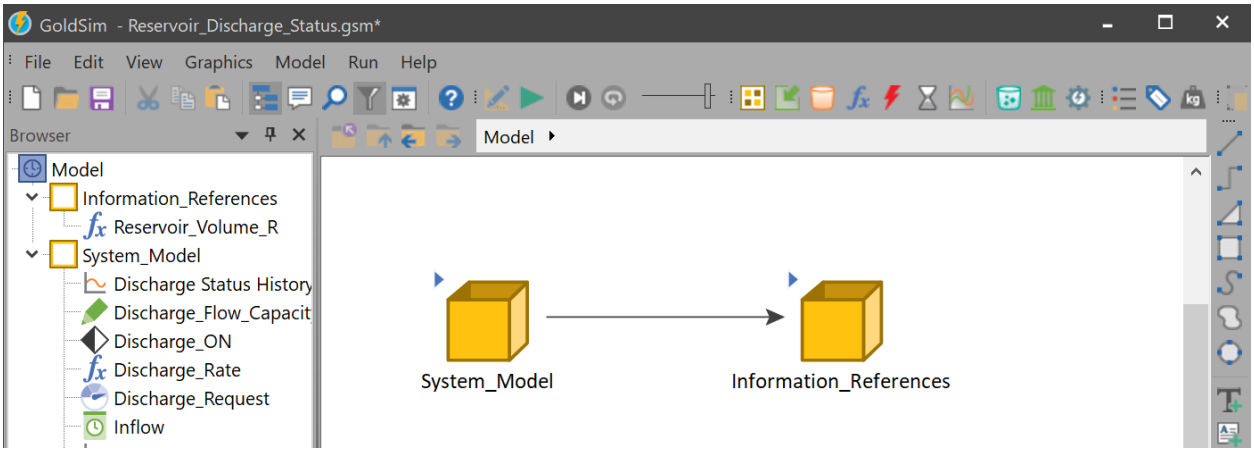
Walk through all the outputs of influence lines in your system model by clicking on each one. As you do so, make note of those outputs that are information only and not referencing a material flow. You will end up with a list of information references.

Organizing Information References

Now that we have identified the information type influence, we can create a reference to the information using an Expression element and give it a name that resembles its special job of being a reference. In the example below, an Expression element is used with a name that ends in the letters “R” and highlighted with a red fill color. This element is doing nothing more than referring to the information output of an element in the System_Model.



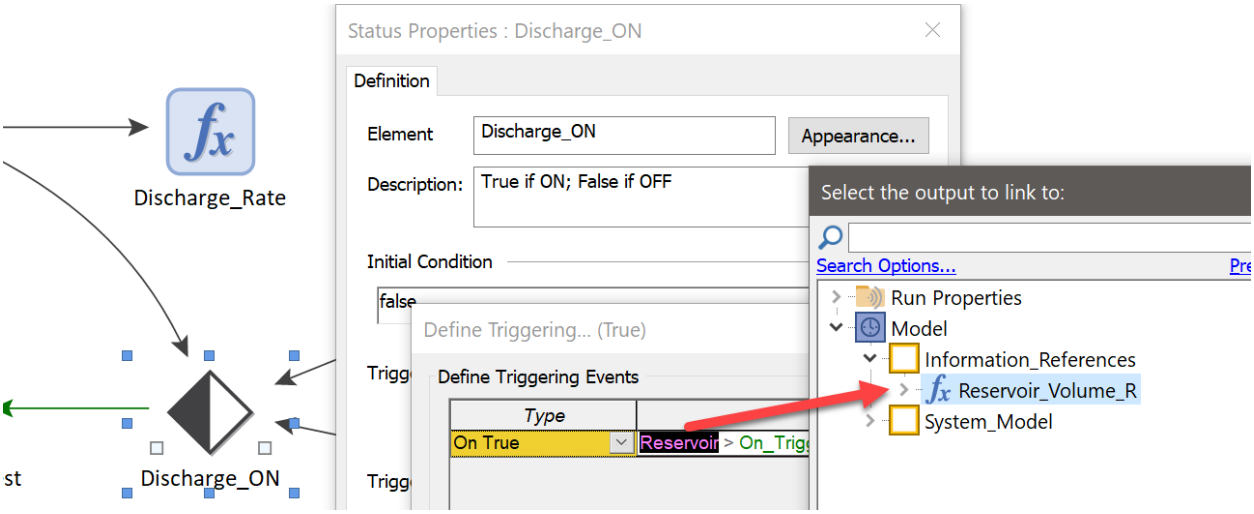
Move this reference element to a new container called “Information_References” then move the remaining elements into another new container called “System_Model”. The top level of the model should now be organized into 2 Containers as shown below.



The contents of the System_Model Container should be further organized into the physical components being represented. For large models, this step is an important part of model construction.

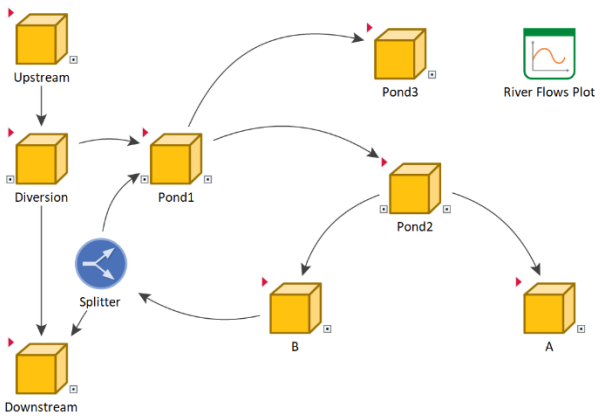
Using Information References

After you move the information reference to the Information_References Container, you should change the element in the System_Model to refer to the external reference instead. In the example shown below, the Status element is currently referring to the Reservoir volume, which causes it to change state during the simulation. This is the element we must change so that it points instead to our external information reference. This will cause the influence line to disappear from the System_Model Container since it is now being referenced in the parent level.

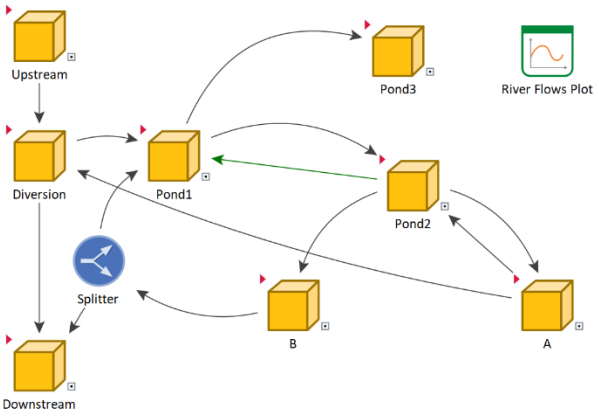


In the examples shown below, showing the graphics pane of the System_Model Container, because the information influences have been filtered out, the only influences remaining represent water flowing through the network. This approach can significantly improve the readability of your system model, particularly for complex material flow systems.

Information References in Parent Container



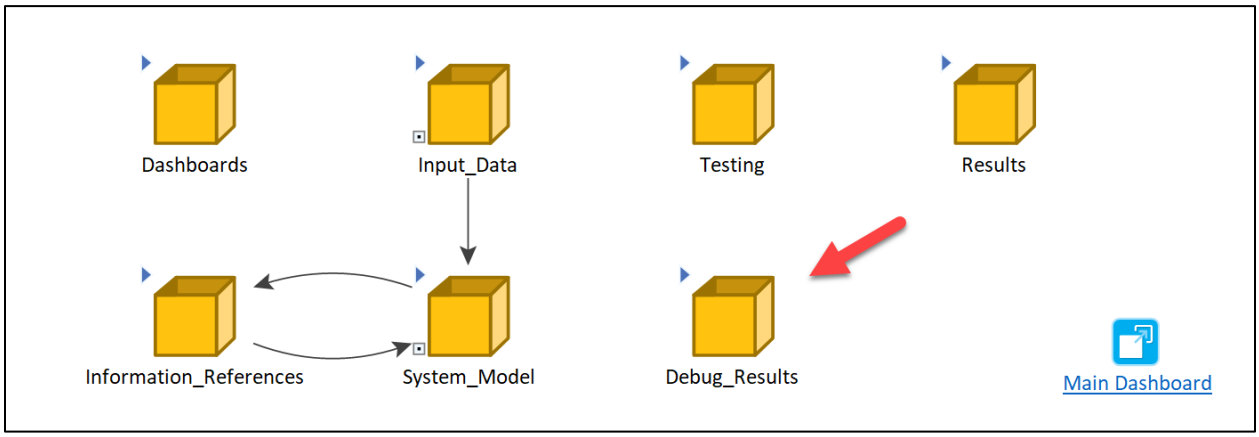
Information References in System Model



4.7 Model Results

A good way to organize summary results for the model is to place them in a top-level Container called "Model_Results". In most cases, it is helpful to organize Result elements into a central location just like you do for the input data so that a user can walk through all the results at once instead of navigating to all the components. A central Container for results also makes it easier to walk through, browse, or search for specific results without having to understand the details of the system.

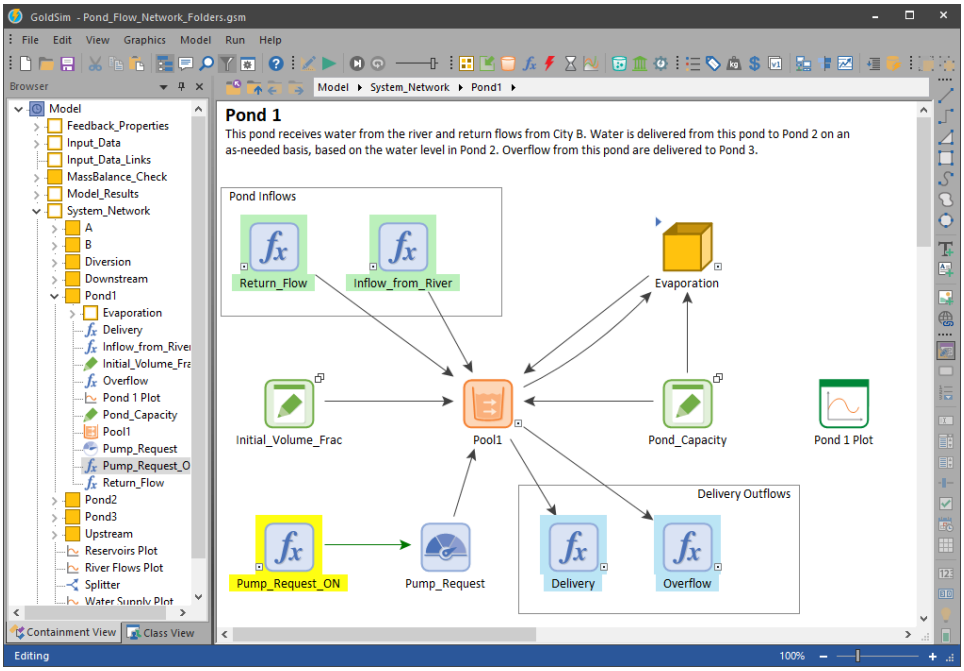
It can also be very helpful to create a temporary Container to store Result elements that are used only for debugging. These are where you store results that help you build the model. If you save these in a separate Container of the model, then removing and disabling unnecessary results at the time of production is a lot easier.



4.8 Element Fill Colors

Element fill colors can be used to highlight certain roles of elements. Examples of these roles might be signaling a later revision to an expression or refer to outside influences.

The example shown below uses an example color scheme to highlight some elements. The example below uses yellow fill for elements that are a work in progress and require the modeler to revisit later for possible revision. Green is used for elements that reference outputs from sibling nodes within the System Container (such as a flow from another component of the system). Elements that provide an output used by other sibling nodes are filled with a blue color. The blue and green colors will give the reader a quick overview of where information is coming from and where it is going.

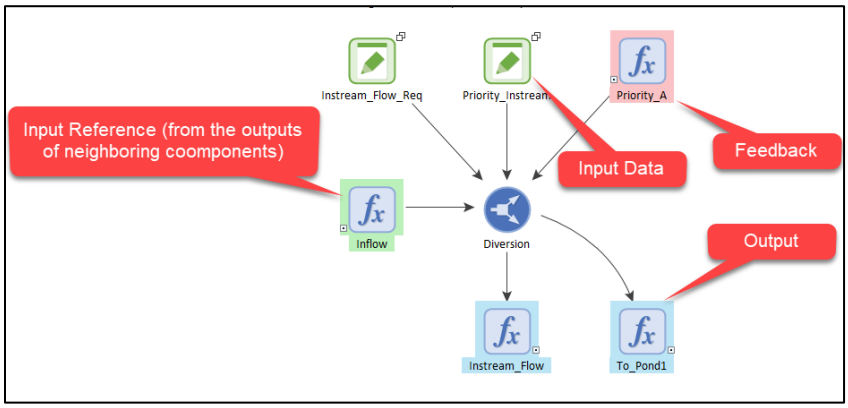


Use bright colors to highlight elements that require revisions so that others will easily see these in the future. Use colors that are not too distracting for other references that are intended to be permanent like for references to inputs and outputs. Table 1 summarizes some color suggestions that you can use in your model, which can be modified to better suit your specific requirements.

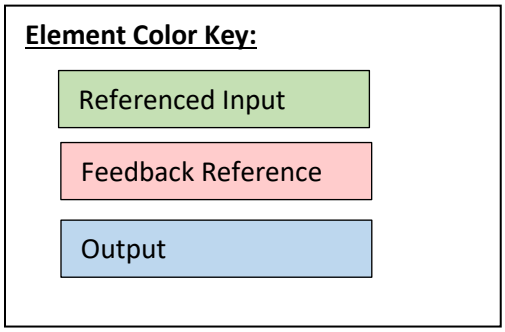
Table 1 – Color Suggestions for Highlighting Special Features of Elements

Special Feature	Color
Work in progress	
Input reference from the outside	
Output being referenced to the outside	

The result is a graphics pane that shows you upon viewing just the colors, where information is coming from and where it is going. The only elements not marked are the local functions used to carry out the process done by the object this container represents.



If you use element fill colors this way, it is recommended that you place a legend in the documentation of the model like this:



5. Testing

You should include tests in the model that ensure it is producing expected outputs. Typically, these tests are run during and after construction of the model. You can store the test logic and results in a top-level Container called, “Testing” or include this within the component Containers of the system. The benefit to storing these calculations in a top-level container is that you can remove them when you put the model into production mode. Some examples of tests are presented in this section to provide you with some ideas, but you should determine what tests are most suitable for you.

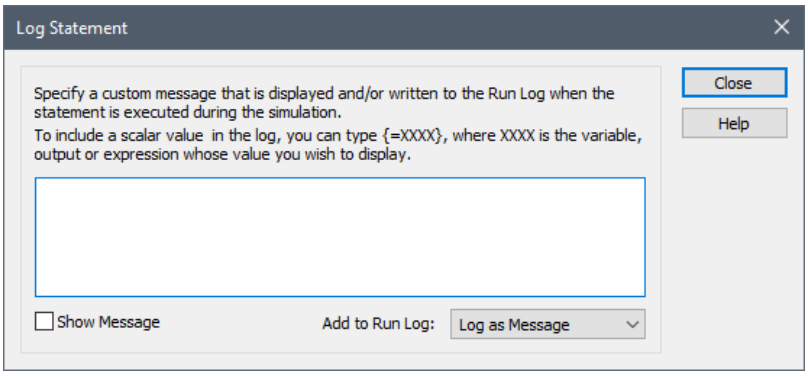
5.1 Verification

Verification testing is the process of evaluating specific functionality of the model to determine whether it operates as designed. Typically, the test will compare the output the model produces to the output from an external source, such as documentation or output from a 3rd party tool. It is recommended that these tests focus on a small portion of the model logic. Like the other tests, these can be organized in a top-level Container.

The Script element can log messages and warnings to the run log of the model. This is a good way to document test results during the testing phase of the model. There are three types of Log statements. All three can be inserted by selecting the appropriate type from the Script element’s Insert menu (or pressing the appropriate hot-key when in the Script dialog):

Log Type	Hot-Key	Behavior
Message	Ctrl+L	Writes a message to the Run Log; however, no warning is displayed after the run.
Warning	Ctrl+N	Writes a message to the Run Log; a warning is displayed after the run to alert the user.
Error	Ctrl+R	Writes a message to the Run Log; treated as a fatal error (the simulation is stopped).

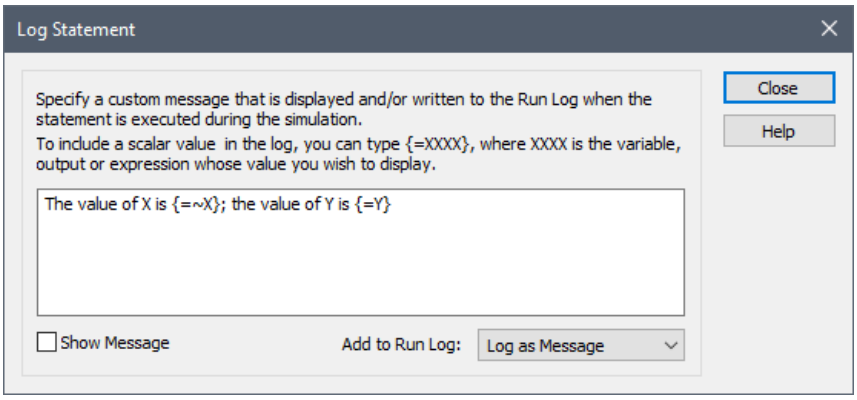
When a Log statement is inserted in a Script, the following dialog will be displayed:



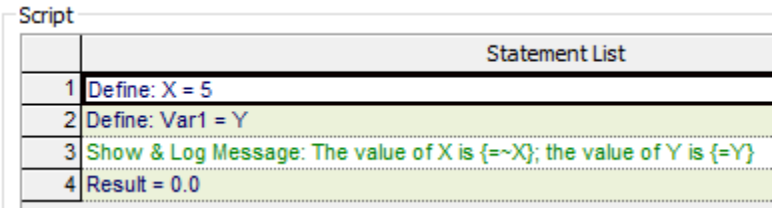
Note that at the bottom of the dialog you can modify the type of Log statement (Message, Warning or Error). If you check the Show Message checkbox, the message will also appear in a pop-up window during the simulation whenever the statement is executed.

One of the most powerful features of the Log statement is that you can write the value of a variable, output or expression in the message. To include a scalar value in the log, type {=VALUE}, where VALUE is the output, variable or expression whose value you wish to display.

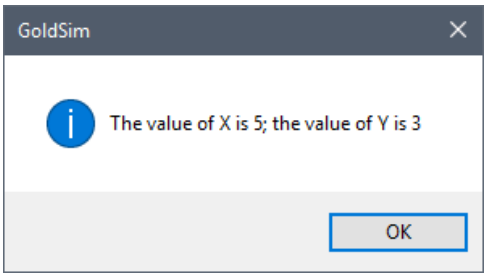
For example, assume you had a script variable named X, and an external output (an output from outside the Script that was referenced inside the Script) named Y. A Log message statement like this:



would look like this in the script:



The message that would be displayed in a pop-up would look like this:

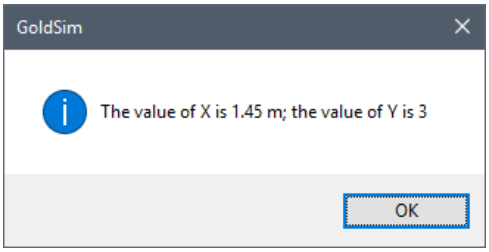


In the Run Log, it would look like this:

```
Realization 1

0 day:
\Script1: Script Message: The value of X is 5; the value of Y is 3
```

If the variables for which you are writing values have units, the units will be displayed. For example, if X was a length, this would be indicated in the message:

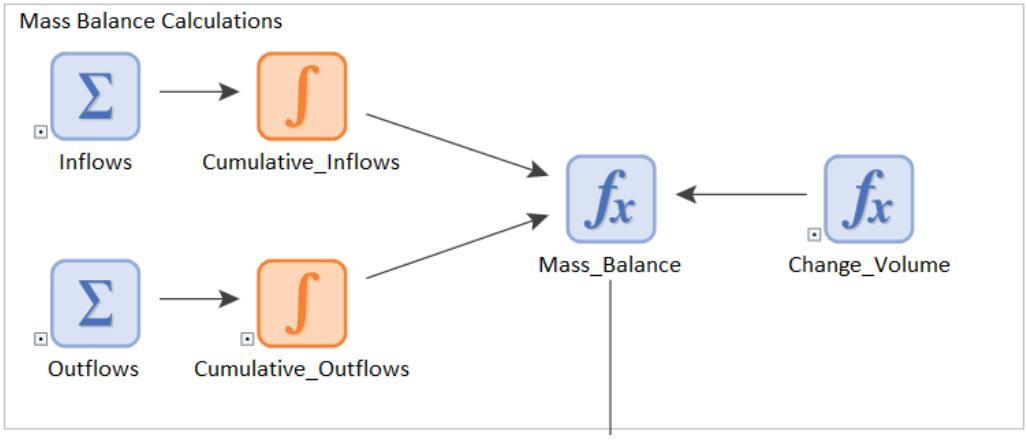


However, *you cannot control the units in which they are displayed*. They are always displayed in the base units for the dimension (e.g., meters for lengths, seconds for time).

The Script element is a good tool to help with testing of the model. Should you want to disable the testing, you could put the statements inside an IF statement that is controlled from a Data element at the top level of the model. For example, you could have a Data element that is set to True or False called "Testing_ON". If true, then all the tests will be evaluated in the Script element. Otherwise, they are skipped.

5.2 Mass Balance

In GoldSim models that represent a system of material flows, it is critical that mass balance testing be performed for the system. This will provide a way to verify that the model accounts for gains and losses in the system. We recommend that you add a Container at the top level of the model specifically to hold the mass balance test calculations and results. An example model that calculates closure of mass balance can be found in our library, here.

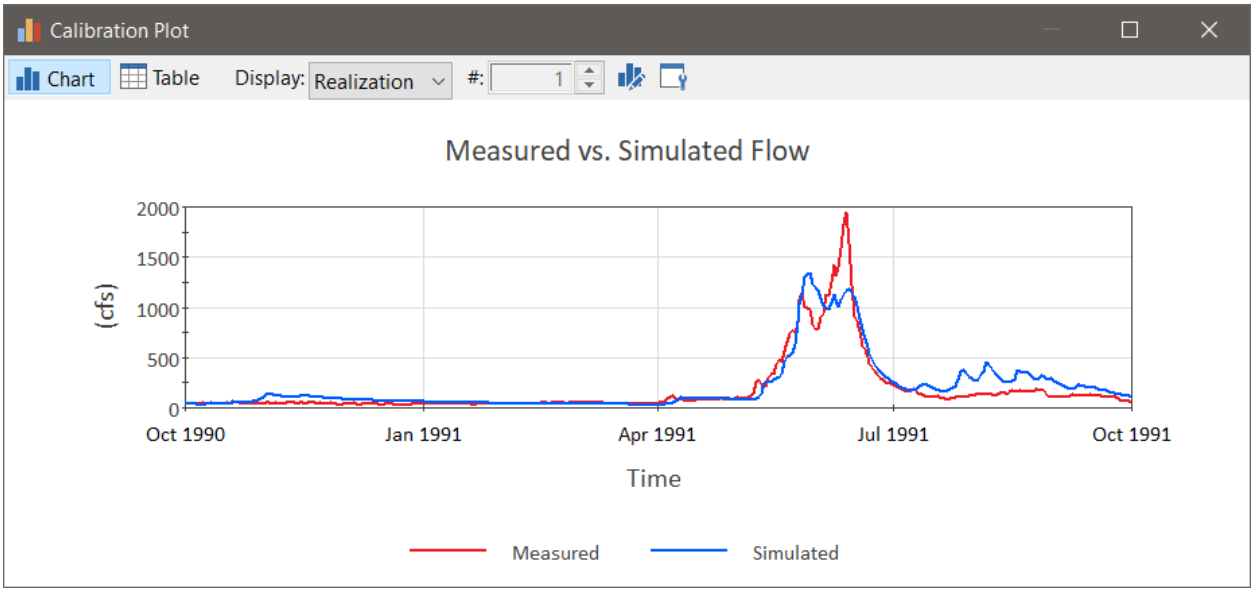


It is also recommended that you include a mass balance check for smaller portions of the overall system. The reason for this is because the mass balance check over the entire system will not tell you where the problem is, should one occur. A test on small portions of the model will point you to the specific portion that has the problem.

If your model is organized in the manner described in previous sections, then linking the necessary flows and amounts for a mass balance test is straightforward.

5.3 Calibration

Model calibration is very important, especially for models that are intended to simulate existing conditions. It is a way to validate the logic in the model and provide a higher level of confidence that it will provide realistic forecasts. It is recommended that a top-level container be added to the model, which includes the calibration information, such as measured outputs that the model is expected to match along with a result plot to show how well the simulated vs measured output match.



You can rely on GoldSim's optimization functionality to test how close a simulated time history matches measured time series data. Examples of this approach can be found in our library here:

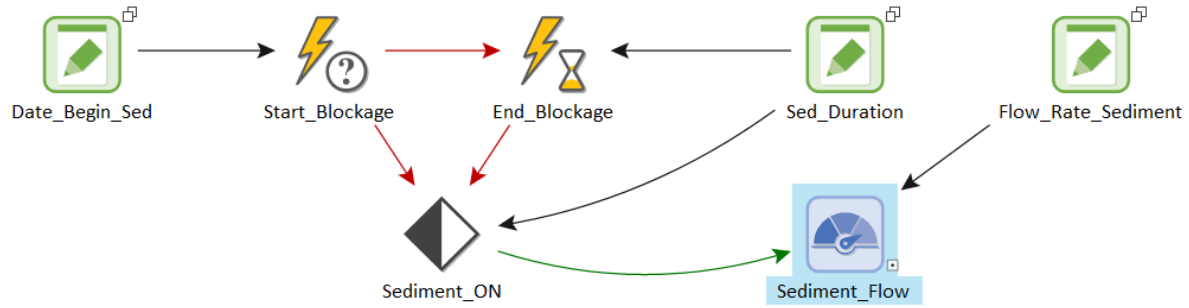
- Model Calibration Using Optimization
- Webinar Archive: Calibrating Your Model
- Root Mean Square Error (RMSE)
- Calibrate a Snowmelt Runoff Model

6. Images and Graphics

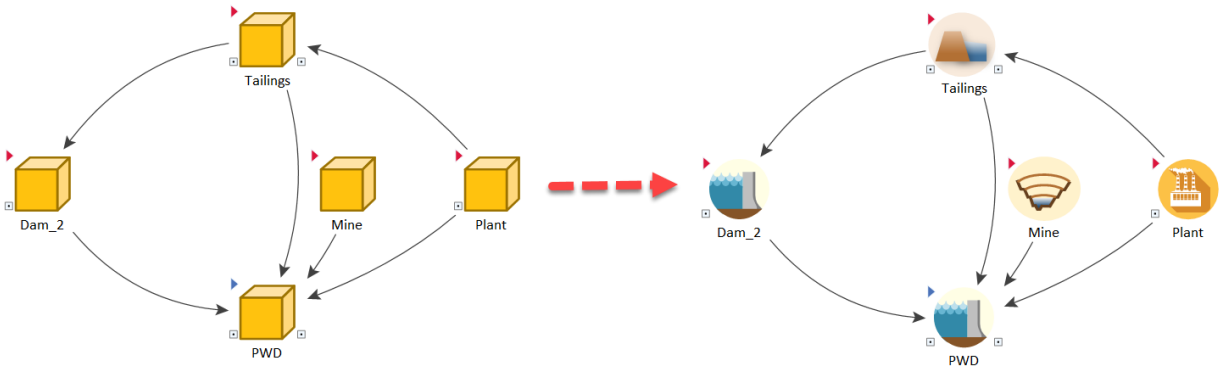
Image files used in your model become part of the model file and therefore will add to the overall model file size. Because of this, you should try to minimize images used in the model. A general rule of thumb should be to use vector drawings when possible and only use raster images when there is no other way (i.e. photography).

6.1 Element Icons

Use GoldSim's default icon and size for all elements except for Containers. The default icon provides context to your model. For example, if a user can see a Sum element it is immediately known what the function is. In the screen capture below, it is helpful to see the default element icons because they provide additional context to the reader. Just looking at the elements will reveal where the inputs, events, conditional status, and the if/then logic are. This provides us with a lot of information without even opening any of the element properties. Replacing these icons with other images would obscure the functionality.

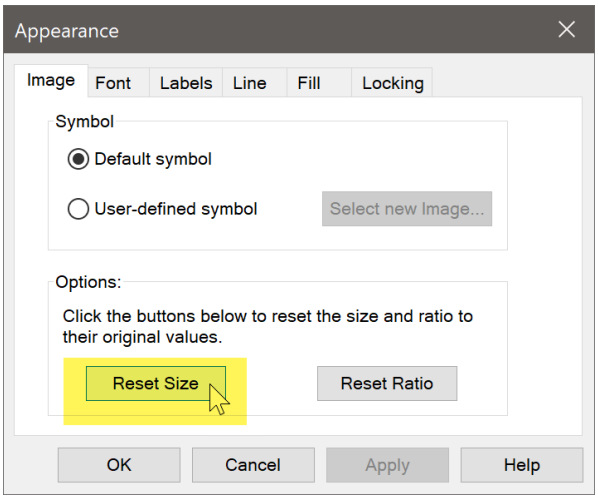


Because you can use localized Containers to represent physical objects within the system you are modeling, custom icons might be useful here. In these cases, you should generate an EMF file for the custom icon by following the instructions here.



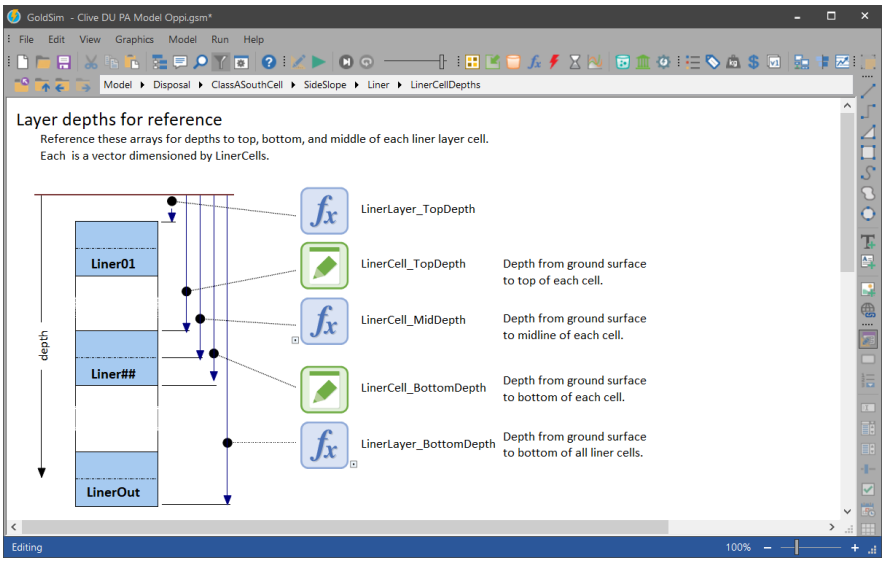
Element Icon Size

In most cases. You should use the default size of all elements. You can reset the size and icon of an element by right-clicking and choosing Appearance then click Reset.



6.2 Embedded Images

Embedding images into the graphics pane can help provide context to the calculations and background to what the model is trying to do. We recommend the use of images to a limited extent. Always choose vector drawings over raster graphics because these are typically much smaller in size and render nicely in GoldSim. Some images like photographs can only be saved in raster format and therefore should be saved at the lowest possible resolution that still looks good but doesn't cause the file size of the model to bloat.



Creating Vector Drawings

The simplest way to embed a vector drawing into GoldSim is to draw it into the graphics pane using the drawing tools provided by GoldSim.

The benefit of doing it this way is that you have the source of your drawing saved with the model so if you ever make changes in the future, it is easy found and edited. The downside is that you are somewhat limited in what you can draw when compared to the tools in MS Power Point or a vector drawing tool.

Importing Vector Drawings

If you decide to create your drawing outside of GoldSim, follow the steps described here.

The recommended steps to inserting new vector drawings into the graphics pane are as follows:

1. Open Power Point
2. Draw the image using Power Point drawing functions
3. Click on the image so that it is selected
4. Right-click on the image and select “Save as Picture…”
5. When the “Save as Picture” dialogue window appears, choose Enhanced Windows Metafile (EMF) from the drop list of file types to save as
6. Save the file
7. In GoldSim, use the Insert | Image… feature to load the image into GoldSim

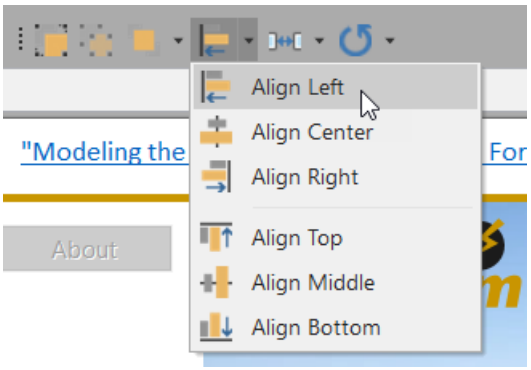
The recommended steps to inserting existing vector drawings into the graphics pane are as follows:

1. Copy an EMF or SVG drawing to the clipboard
2. Open Power Point
3. Paste the image
4. Right-click on the image and select “Group” → “Ungroup…”
5. You will be prompted to convert the drawing. Click Yes. Now it becomes an EMF that will render nicely in the graphics pane
6. Right-click on the image and select “Save as Picture…”
7. When the “Save as Picture” dialogue window appears, choose Enhanced Windows Metafile (EMF) from the drop list of file types to save as
8. Save the file
9. In GoldSim, use the Insert | Image… feature to load the image into GoldSim

For more information about how to embed images into GoldSim, please refer to this article: File Formats of Images Inserted into GoldSim.

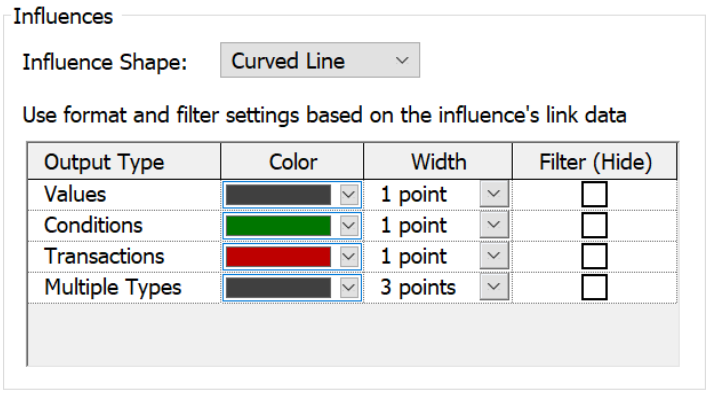
Alignment Tools

Use the alignment tools in the graphical toolbar to align images if needed.

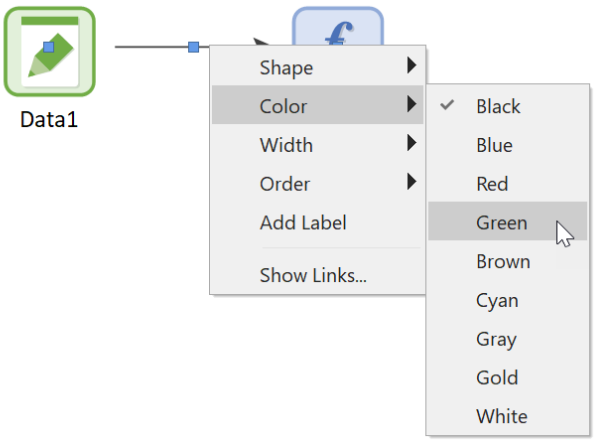


6.3 Influence Line Colors and Shapes

Use the default global influence line colors and shapes for the different types of information, as seen in the Container properties dialog.



If needed, modify the color of influences to highlight key relationships. Colors can be used to identify the type of information flowing, and in some cases this can be useful. To change the color of a single influence line, right-click and choose color.



7. Memory Used

The goal for your model should be to use no more memory than what is needed to portray results of your model and understand what it is doing. You might need to save more results temporarily while debugging but that should not be considered the ultimate state of your model. This means you need to save result elements only used for debugging and building the model in a separate place (as mentioned in the “Model Organization” section earlier) so you can easily remove them and disable the results later on.

A rule of thumb for a model is to start building the model assuming no outputs need to be saved. Only turn on outputs that you need as you continue building the model. With this approach, you minimize saving results that aren’t needed. The opposite approach is to start out by saving everything by default then deciding at the end to disable what is not needed, which ends up taking more time and causing unnecessary bloat.

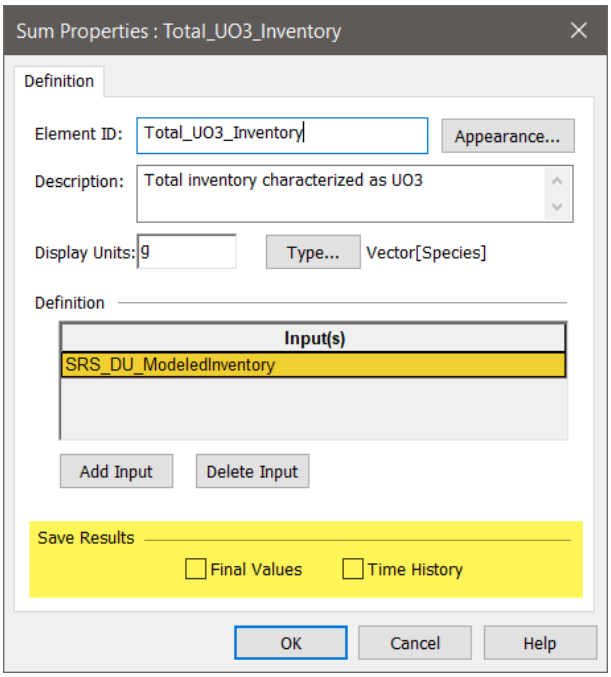
The memory used to view and run a model is a function of the following model properties:

- Number of (and type of) elements
- Number and size of arrays
- Amount of text in element notes and text boxes
- Amount of data stored in time series and lookup tables
- Graphical components
- Number of update time points for saving results
- Number of unscheduled updates occurring
- Number of realizations and scenarios
- Number of external elements, their DLL code, and the data

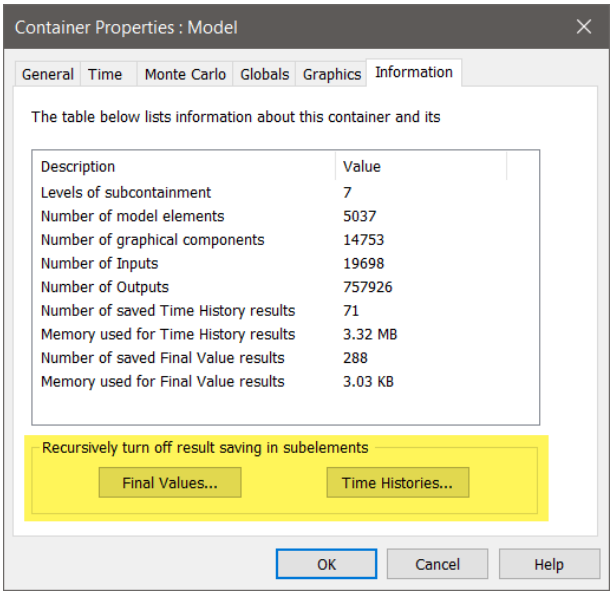
The memory used is an important aspect of your model because it will affect the size of your model file, which needs to be shared with others. If the size of the model file and results continues to grow as you build the model and you don’t take any measures to restrict the size, it’s possible that you approach the limits of your operating system. To avoid this, we recommend that you take memory reducing measures while building the model and also take some time before publishing your model to disable saving results for outputs that are not needed while the model is in production mode.

7.1 Disable Final Values and Time History Results

The check boxes at the bottom of an element’s property dialog control whether outputs will be saved for the element.

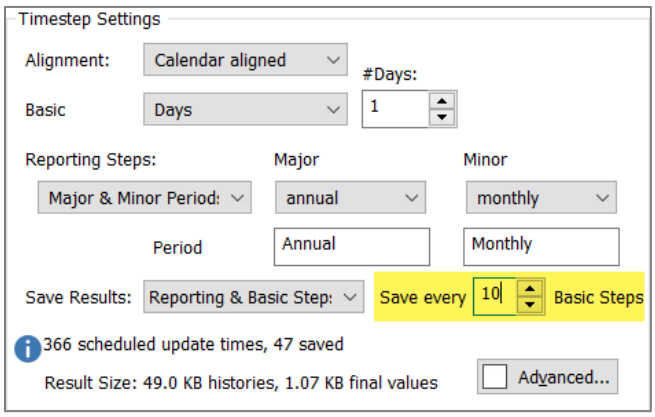


You can set this save setting globally by right-clicking in the top level graphics pane and going to the Information tab. In there, you will see two buttons that will disable all final values and time histories. This is an easy to way to start fresh with an existing model then walk through and only enable what is needed.



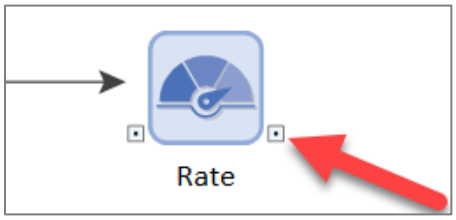
7.2 Time Step Settings

Start by assuming a larger time intervals for saving results and only reduce the interval when it becomes necessary to see more time points. You can find this interval setting in the model's simulation settings dialog.



7.3 Delete Unused Elements

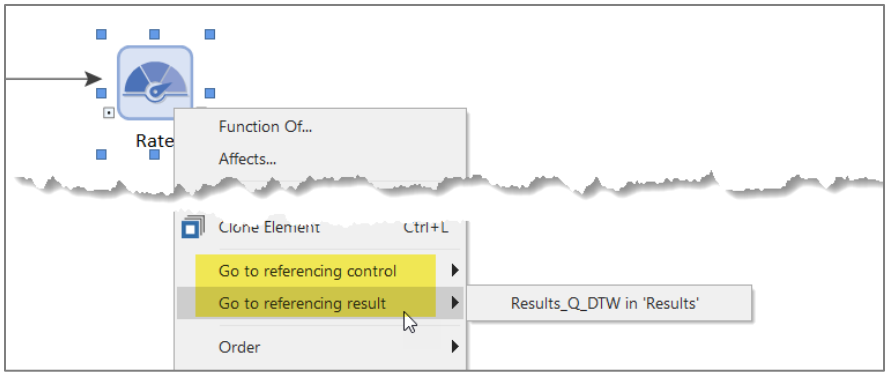
It is often helpful to walk through the model before going into production to make sure you don't have any loose ends where elements are not being used anymore. This happens when you are building the model and change how elements are linked to each other during further iterations of model changes. Before deleting an element, make sure there are no dependencies remaining. To do this, first inspect the output port. If you see a "dot" inside the port, it means other elements are referencing it. Deleting this element would cause the model to be in an error state, unable to run.



There are more subtle dependencies that you need to be aware of:

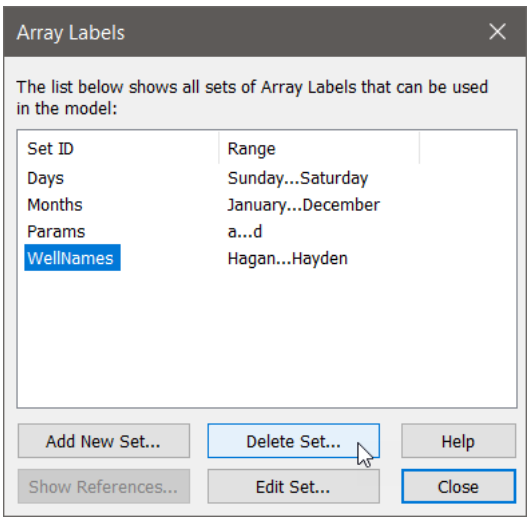
- Result references
- Links to Dashboard controls

To check for these types of dependencies, right-click on the element and check for referencing results and/or Dashboard controls. If you see either of these two items in the context menu of the element, it means there are dependencies that will be destroyed when you delete the element. You should first resolve these by linking the results and/or Dashboard controls to the correct element before deleting.



7.4 Delete Unused Array Label Sets

During the construction of a model, you might create array label sets and stop using them later on, causing an accumulation of unused array label sets in your model. Walk through all the labels and click the Delete Set… button to remove any unused sets.



8. Documentation

A model which cannot be easily explained is a model that will not be used or believed. As a result, GoldSim was specifically designed to allow you to effectively document, explain and present your model. You can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences.

The most important way to ensure that your model is easy to explain and present is to spend some time to organize your model. Use the ideas presented in the previous sections to better organize your model.

The goal for documenting a GoldSim model should be to provide all documentation needed for the reader to understand the model without having to refer to an external document. It's expected that you might need to refer to external documentation of any well-established functions used in the model, which provides some needed context.

8.1 Display Units

It should be decided at the start what units are used for various types of calculations in the model. You should avoid building a model with a mix of display units. For example, half of all flow type elements have display units of ML/week and the other half uses m3/d for no apparent reason other than one person had a different preference at the time or because nothing was decided. The result of this is that you must change a lot of display units in various elements or charts to maintain consistency throughout the model.

8.2 Text Blocks

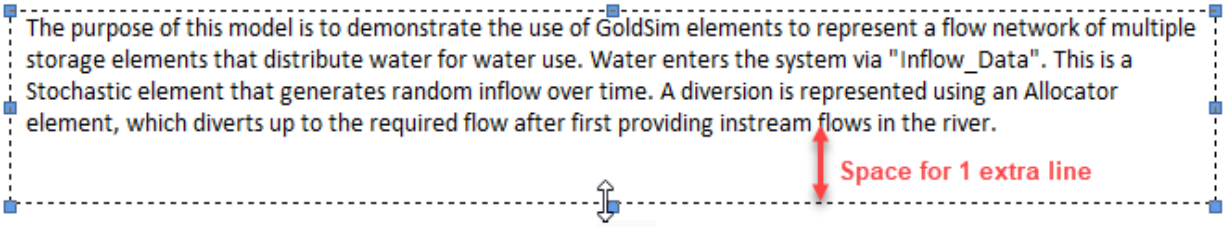
GoldSim provides two ways to add text to the graphics pane:

- Text object
- Text Box

Text objects are movable objects that can utilize a single font. Text Boxes are scrollable boxes of text that can simultaneously utilize multiple fonts and can be "pinned" to a specific location in the graphics pane. We recommend that you use Text objects unless a special formatting is needed like hyperlinks, sub/super scripts, bold sections, etc.

Use the default font for text in the graphics pane except for headings. The default font is Calibri 11.

Make the bounding box large enough to allow the text to grow if viewed on a screen with different resolution or different zoom scale. The rule of thumb is to provide an extra line of text.



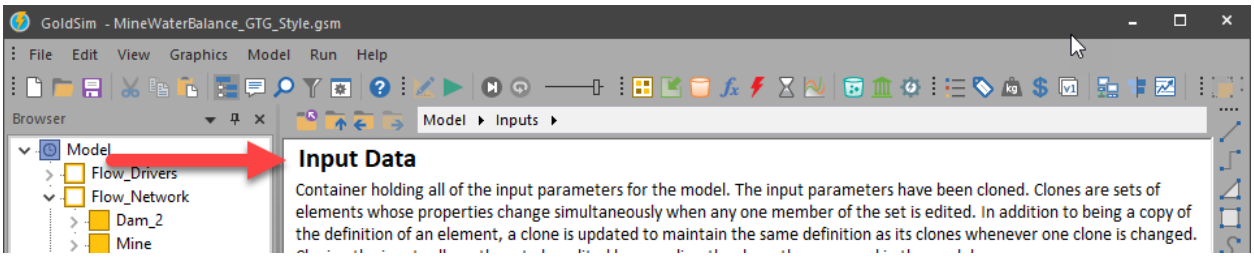
8.3 Element Descriptions

Use the Description field for all elements in the model, including Dashboards and Results. Because the Description is displayed in tool-tips, it can be an effective and highly visible documentation tool.

Start with a capital letter on the first word of the description. Try to contain the description within the 2 lines provided. If you need more space, abbreviate then put a more descriptive version in the note pane.

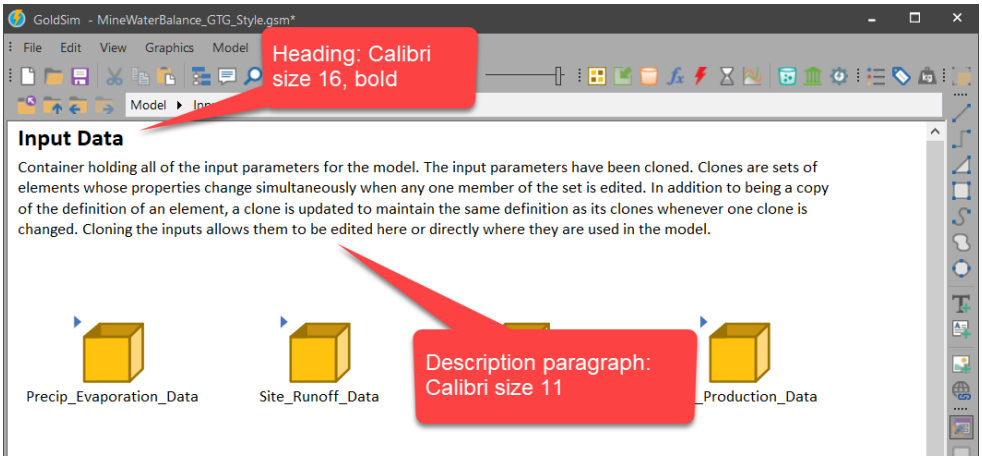
8.4 Container Heading

Every Container in your model should have a title. Put a heading at the top left corner of the graphics pane of every Container, written in plain text using Calibri bold, font size 16. Make sure the text block is moved to the top-left corner of the graphics pane.

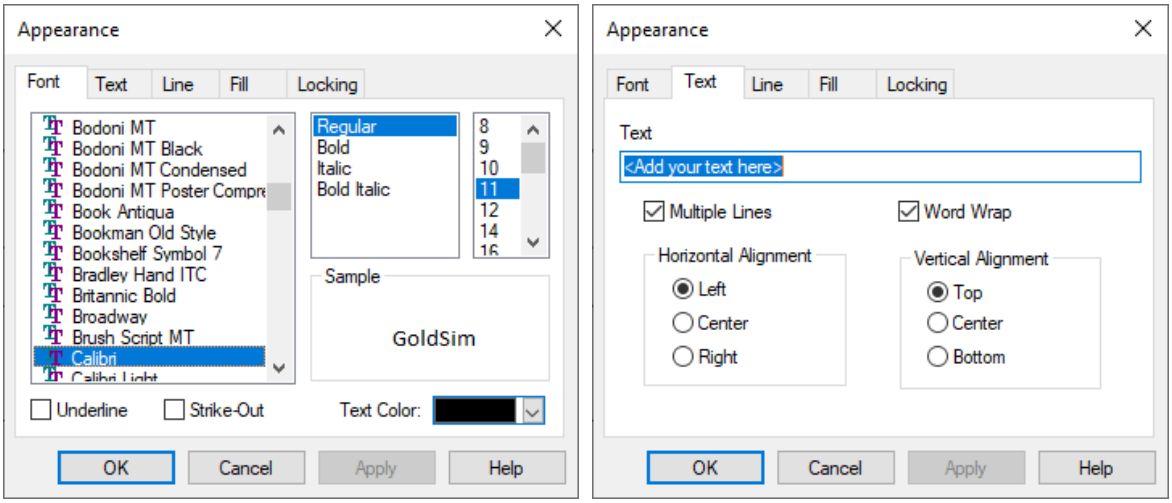


8.5 Container Description

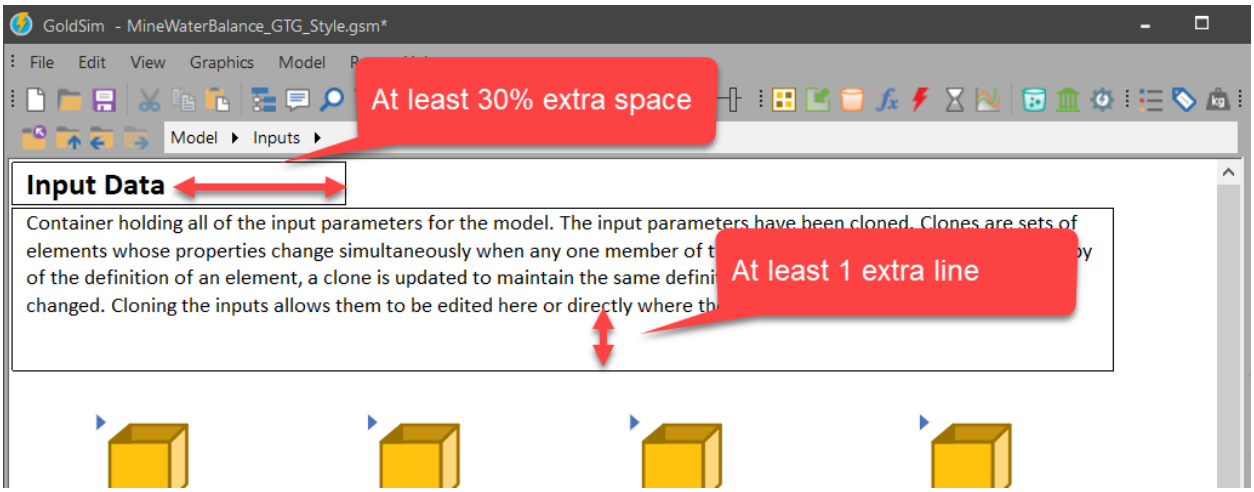
Every container should have a short description located at the top, just under the heading. Omitting text here leaves the reader with the task of opening elements just to figure out what is happening. Add a short paragraph just below the header to provide a basic description of what is being done in this container. Make sure the text block is moved to the top-left corner of the graphics pane, just below the heading. It is recommended that you keep the width of the text block to less than 8 inches.



For reference, the font settings for all plain text in GoldSim should have the settings shown in the properties dialog below.



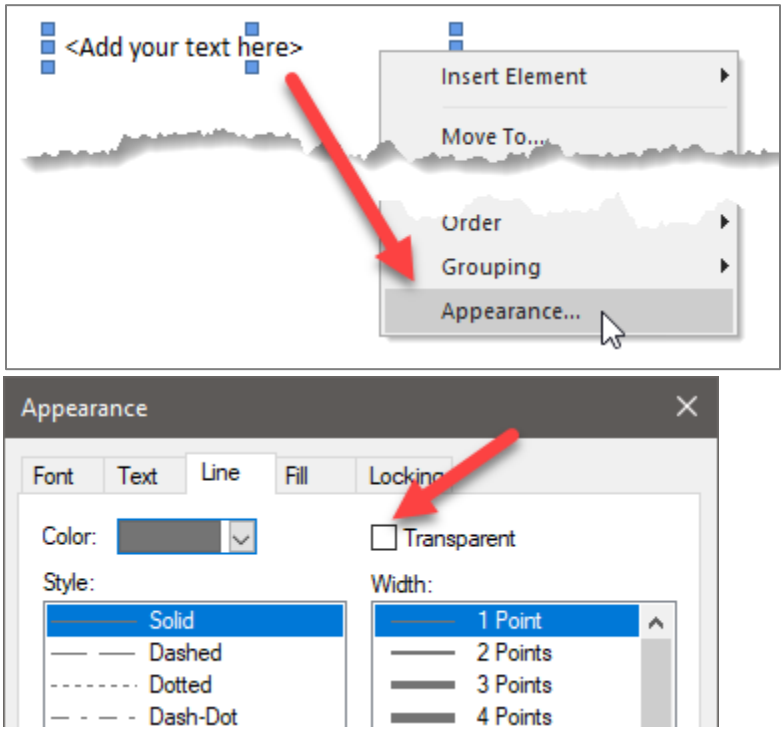
Because text is rendered differently depending on the resolution and Windows screen scaling, you should provide some extra room for the text to grow within the box. The screen capture below shows the same text but with the text border turned on so you can see how much extra space is provided.



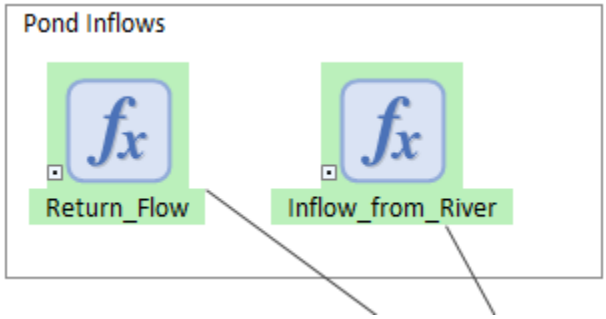
If more space is needed to further describe what is happening in the container, consider adding a child Container for more documentation. Alternatively, you can add more text at the bottom of the graphics pane below the elements. You should avoid adding a wall of text that requires the reader to scroll down the page to find the model elements.

8.6 Section Borders

Use a plain text block to draw a bounding box around parts of your model. Turn on the text box outline to show the box. Do this by adding plain text and changing the outline to have a color.



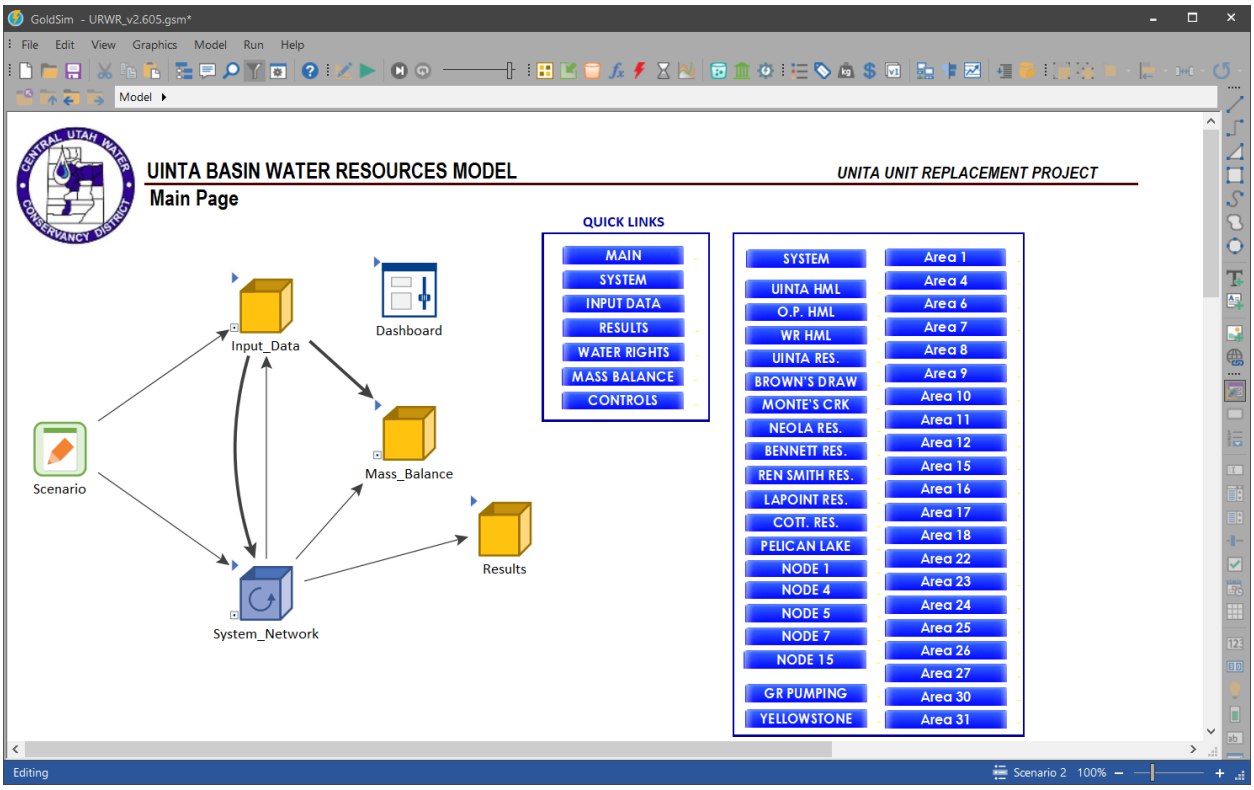
The result is a single object that shows a box along with the heading:



8.7 Hyperlinks

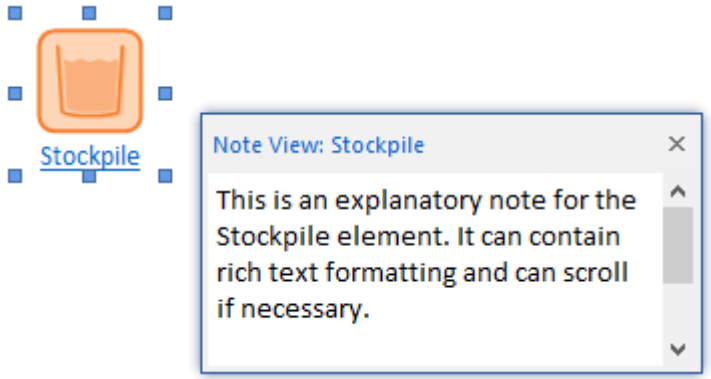
You can use the Hyperlinks to provide navigation aids for your model. The Hyperlink can be used to provide links “jumps” to specified Containers, Dashboards or elements from any location. The links

shown in the screen capture below are using image files as a background to make them look like buttons.



8.8 Notes

We recommend that you add notes to any elements where two lines in the description field is not enough. In order to internally document your model, GoldSim allows you to attach a Note to each element:

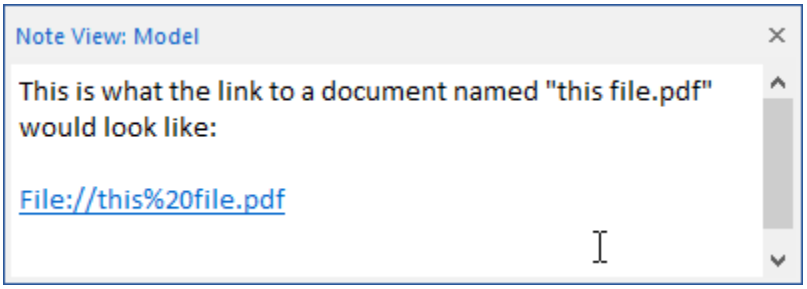


An element with an attached note has its ID underlined (and by default the text is blue). You can modify this so that the ID is shown in the defined text color (black by default) via an option on the General tab of the Options dialog (accessed by selecting Model|Options from the main GoldSim menu).

One of the most powerful features of Notes is that you can insert hyperlinks to other documents:

- To insert a link to a URL which starts with "www", you need only type in the address (e.g., www.goldsim.com).
- To insert a link to a URL which does not start with "www", you must specify the transfer protocol (e.g., http://website.com/).
- To insert a link to an ftp site, you need only type in the address (e.g., ftp.asite.com).
- To insert a link that launches an email, you must type “mailto:” before the email address (e.g., mailto:name@company,com).
- To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you must type "file://" before the document name the full or relative path. Excluding the path points to the current directory of the model file.

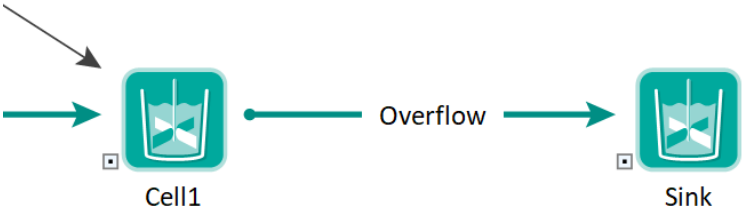
When you are entering a hyperlink, GoldSim recognizes it as a link as soon as it sees a space (it will become underlined and the text will become blue). As a result, when you are entering the text, there cannot be any spaces in the link. If the filename includes a space, you can represent this by using the characters %20 to represent the space. For example, if the filename you wanted to link to was “this file.pdf”, the link in the Note pane would need to look like this:



When you click on a hyperlink in a note, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

8.9 Influence Line Labels

For some complex relationships among elements, you can add labels to influence lines. Influence line labels are particularly useful for describing the inflows and outflows driving Cell Pathway mass transport.



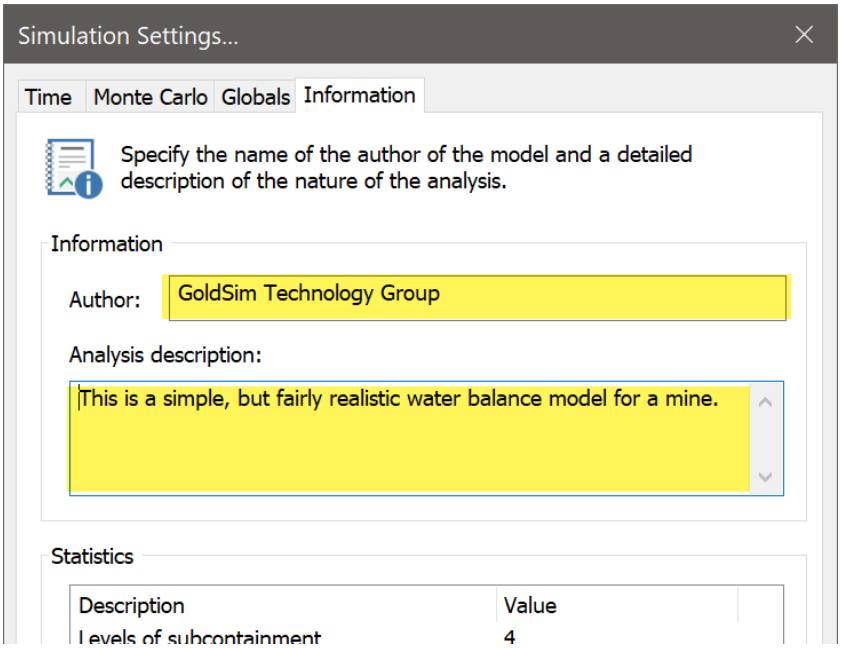
You can add text to an influence by right-clicking on the influence and selecting Add Label from the context menu, or Shft+double-clicking on the influence (double-clicking on the influence while pressing

the Shft key). This will add a textbox to the influence with the word "Text" inside (which you can subsequently replace with your own text).

Keep in mind that if you move elements (and their associated influences) to a different container then the labels will be lost.

8.10 Simulation Information

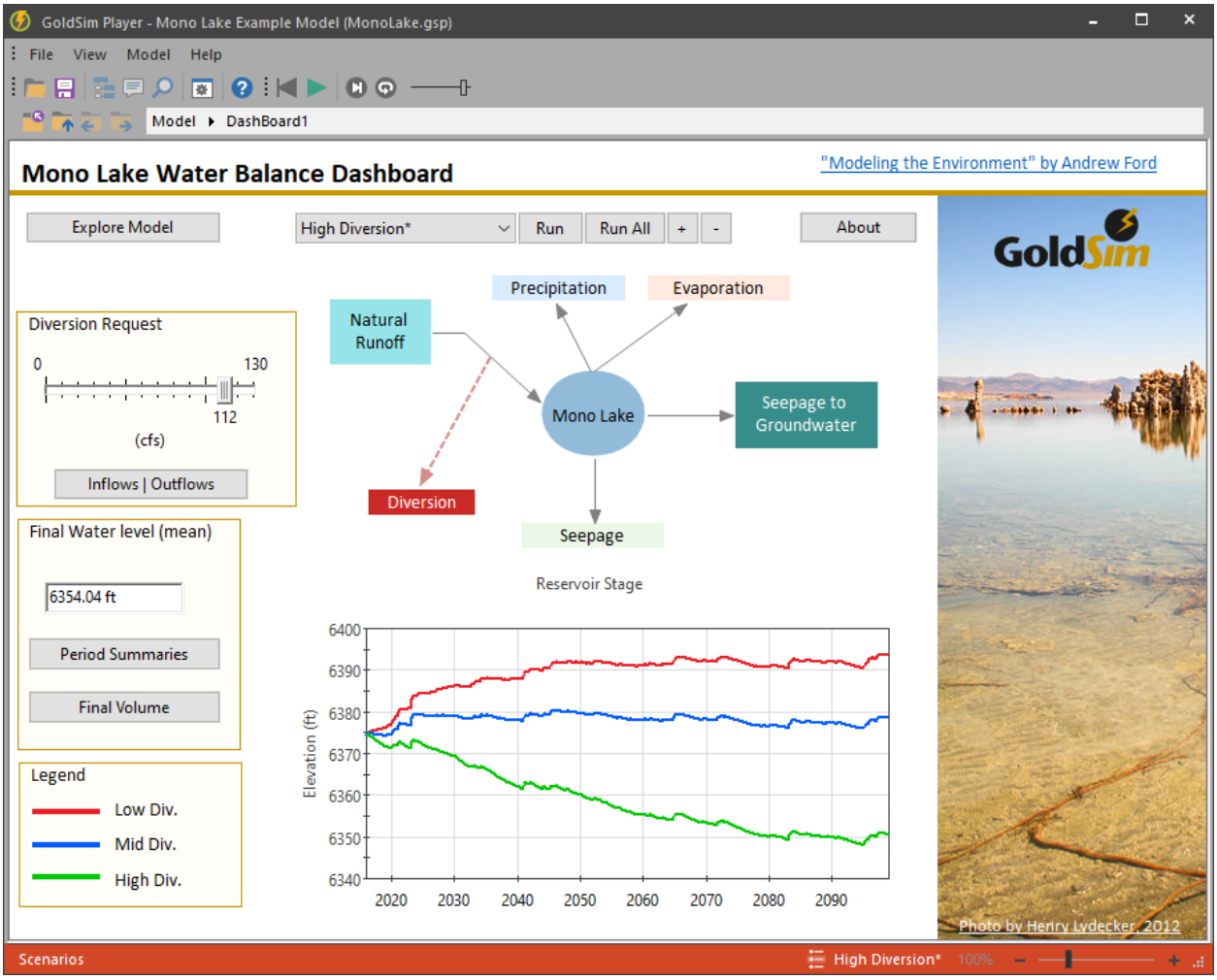
Provide the name of the main author or the organization in charge of building the model along with a short description of the model in the Information tab of the Simulation Settings.



The model author is automatically displayed in the run log. You can also choose to show this and the description in the footer of a result chart.

9. Dashboards

We do not provide very many specific style recommendations for Dashboards because these are highly customizable and depend on the requirements of the model and your audience. The most important qualities of your dashboard are that you portray just what your audience needs to evaluate the system and understand the results. It might require multiple dashboards to accomplish this.

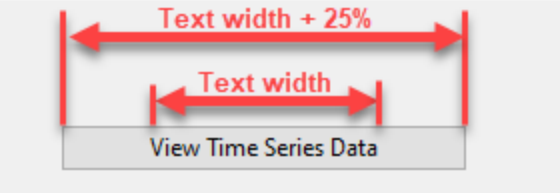


9.1 Colors

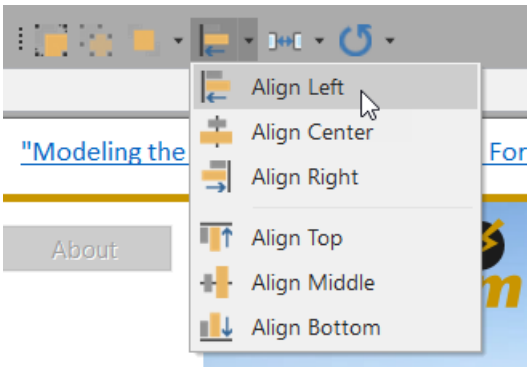
It is recommended that you develop a color scheme for all dashboards in your model and stick with it. Typically, this will involve a very light color for backgrounds and more color saturation for accents like borders and dividers.

9.2 Sizing and Alignment of Dashboard Controls

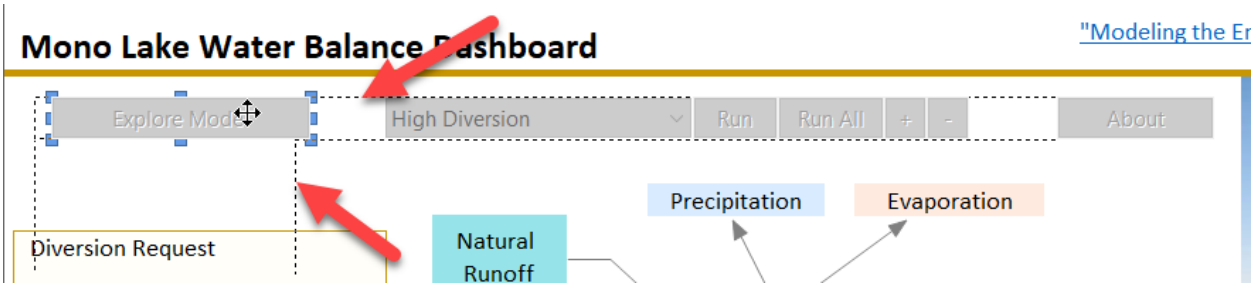
It is very important that the controls on your dashboard allow for different screen resolutions and Windows scaling factors. To accomplish this, we recommend that you allow for 25% growth of the text within controls.



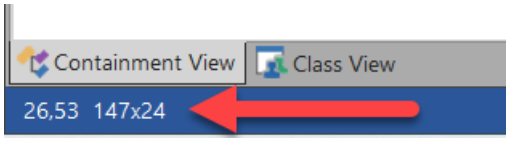
Use the alignment tools in the graphical toolbar to align controls.



When you move controls, you can see that GoldSim provides alignment functionality with guidelines and snapping ability that will guide you in alignment as well.

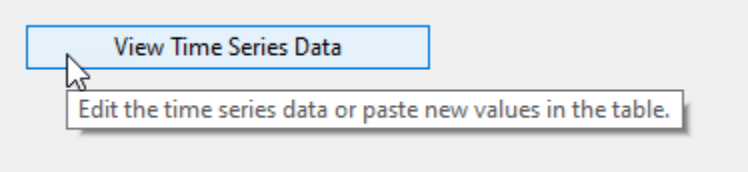


For more refined control, you can view the x,y position and size of controls displayed in the left side of the status bar:



9.3 Tooltips

Add tooltips to all dashboard controls to give the user added instruction about the control. It might seem redundant and unnecessary to those building the model but typically the end-user of a Dashboard will need more context about the controls.

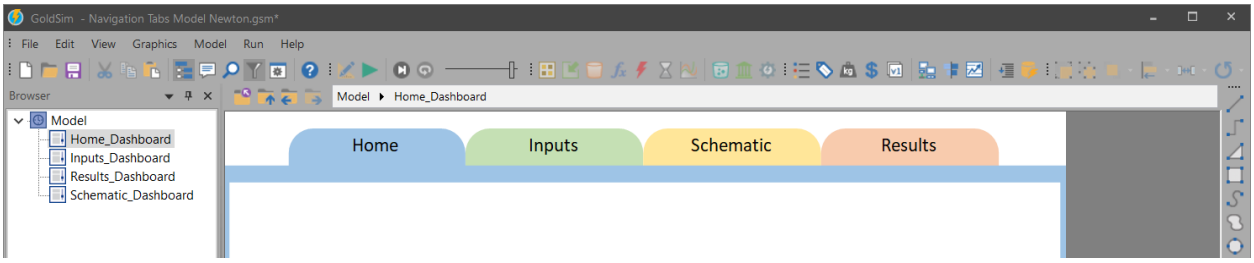


9.4 Dashboard Hierarchy and Navigation

Some models require more than one Dashboard. In cases like this, you may need to provide some structure to the navigation between Dashboards. The simplest way to do this is using a series of buttons that open other Dashboards. The same series of buttons should be copied to all Dashboards of the model. The button that opens the currently viewed Dashboard will not do anything and you should color this button differently from the others. You can align these navigation buttons at exactly the same point on each Dashboard so they appear stationary to the end-user while they navigate.

Below is a screen capture from a Dashboard called “Main Home”.

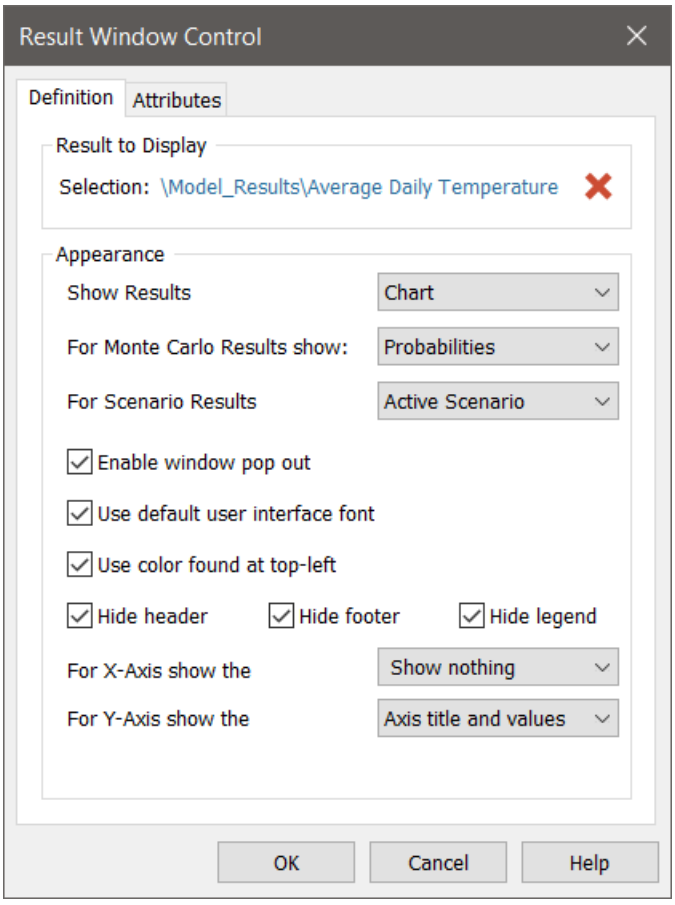
Another option that is more complex to build is buttons that provide the appearance of navigation tabs as shown in the example below.



If you want to add navigation tabs to a Dashboard, you will need to create multiple Dashboards with graphical components that align in order to give the impression that you are switching tabs instead of switching entire Dashboard pages. Luckily, with a few simple steps, this is quite easy to do. Follow these basic steps to quickly create your navigation tabs for your Dashboard.

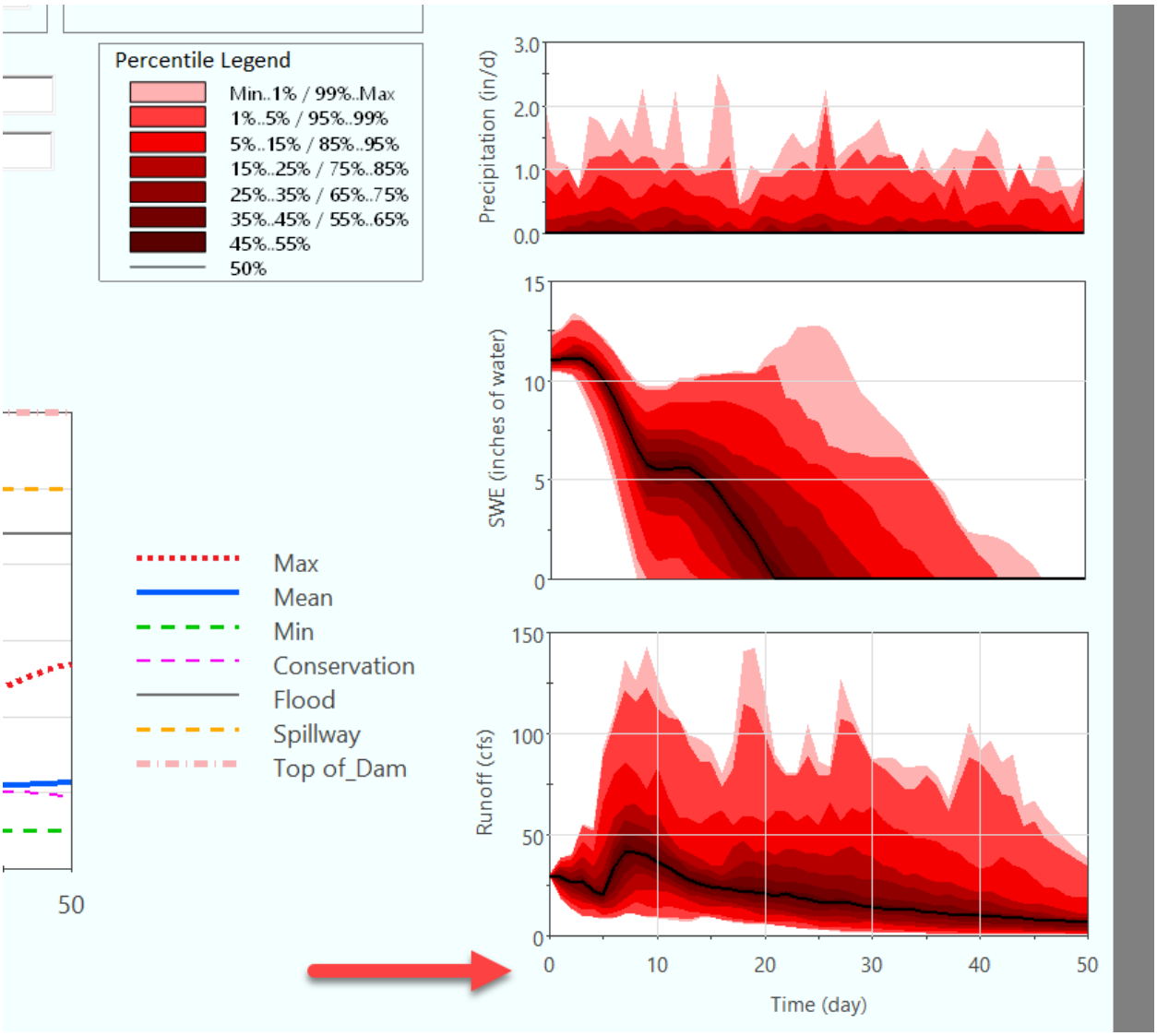
9.5 Embedded Charts

There is a lot of flexibility in how you show embedded charts on Dashboards but you should use a consistent style and stick with it. Our recommendation is that you use the default user interface font so that it matches the font of other controls in the Dashboard. You should enforce consistency in the appearance overrides of the embedded chart:



We also recommend that you do not show more than is needed to portray chart information. For example, you might have multiple charts showing on the Dashboard and if they are aligned vertically,

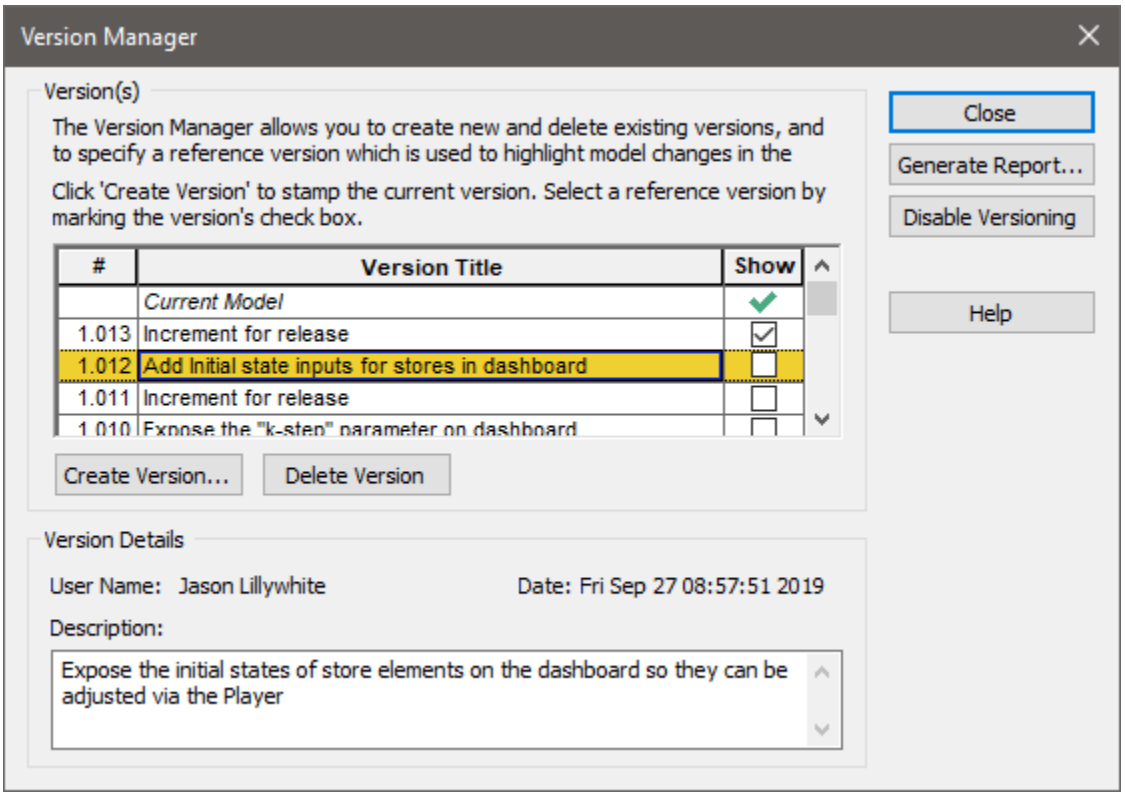
then only the bottom chart needs to show the x-axis. In the example shown below, two of the three charts are not showing the x-axis.



10. Versioning

GoldSim provides the ability to track changes that you have made to your model file. This feature (referred to as versioning) allows you to quickly determine the differences between the current version of your model file and some previous version of the file. Providing this configuration management capability is particularly useful for:

- Coordinating model changes when multiple people can access and modify the model file;
- As a Quality Assurance/Quality Control feature enabling you to demonstrate and document when and what changes have been made to a model file.



We recommend that you carefully plan each change to the model before starting. With this plan in place, increment the version number and write a short description of what you plan to do then make the changes. If you are sharing the new version of the model with someone else to review, append the version number to the model file name.

Don't try to do too much at once. It is recommended that you take on one task at a time and break complicated tasks into multiple steps, each with their own version increment. For large teams working on a single model, we recommend using a check-in/out process like the one provided in this example.

Keep in mind that you can edit the version description after you create that new version, so this allows you to go back and add detail after making some changes. Typically, it is not fully understood what the changes will involve until carrying them out.