GroveStreams Modeling Guide

Table of Contents

Getting Started

GroveStreams works with streams of data. A stream of data is usually a collection of values, each associated with a time or time range. Modeling tells GroveStreams how to manage each stream of data and extract critical information so that your data can be viewed or acted upon as it arrives.

Modeling involves:

- Associating each stream with a GroveStreams organization
- Associating your stream with metadata that describes how the stream is to be viewed. Metadata examples include units, the number of decimal places and what type of default chart should be used
- Defining GEO location streams
- Defining virtual streams based on rollup calendars
- Deriving streams from:
    - Other organization streams and spreadsheet type formula expressions
    - Internal and External RSS feeds such as weather or currency exchange rate feeds
    - The aggregation of thousands of other streams
- Defining complex events, actions and notifications

Components

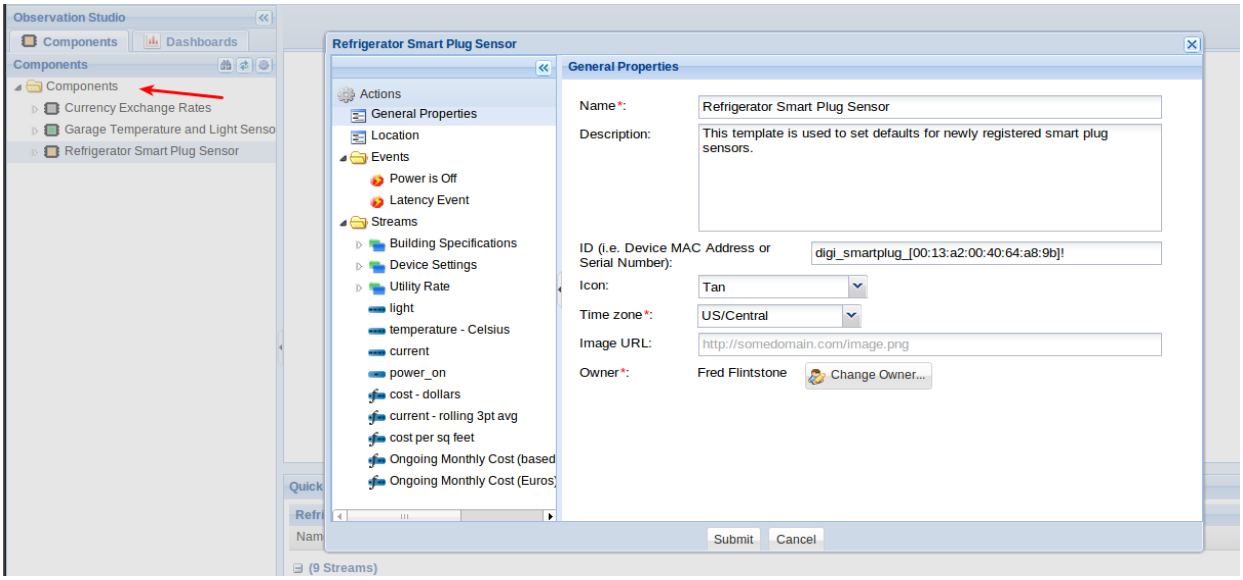
Streams and the components they belong to is where most modeling takes place. Let's start with a Component.

A component is a "thing" that is also a container for a group of streams and events that share a similar location. For example, a component might be a sensor that monitors temperature and humidity. Temperature would be one stream and humidity would be another stream. If the sensor were mobile, its component could also have latitude and longitude streams that record the location of the sensor component over time.

Components and Streams need to be modeled in a manner that ensures compatibility with the streams of data arriving.

Creating a Component

Right click on a folder in the Component browser, located on the left side of Observation Studio, and choose **New Component**. The Component editor window will appear.



General Properties

- Name: Give your component a name. The name is used throughout the GroveStreams (GS) user interface. The name can be changed anytime.
- ID: Optional. This ID can be included on Feed JSON PUT calls to associate parts of a feed with this component. Component IDs have to be unique across an organization. It is highly recommend that component IDs are used within Feed API calls. The ID can be changed anytime. GroveStreams will associate an internal id, referred to as a UID, with this component when it is saved. UIDs are used internally and can be used in Feed PUT calls but are not recommended.
- Icon. Select an icon that represents this component. The icon is used throughout the UI including within dashboard maps.
- Time Zone. Select a time zone for this component and all of its streams. The time zone is important as it is used by several time calculations throughout GS. For example, rollup calculations will use the time zone to determine the start and end datetimes of each cycle such as the start and end datetimes of day or month cycles. When viewing datetimes in the GS browser UI, the user's configured browser's time zone will be used for displaying purposes.
- Image URL: Optional. Enter an image URL for the component. Currently, the image is only displayed in Component views.

- Owner. Select the owner of the component.

Location

All location information is optional except for the Disposition.

- Location Description. Optional. The described location of the component. It could be a rack number, a general location description, such as 'in the SW corner of the warehouse', or any other location description.
- Address. Optional.
- GPS Coordinates.
    - Longitude and Latitude Geo streams are required if the component's disposition is defined as mobile. Altitude is optional.
        - Geo streams need to be created and modeled for mobile components. They are not created automatically. Geo streams are just like other streams except that their data types are set to LONGITUDE, LATITUDE, or ALTITUDE.
        - Geo streams can be regular, point, or interval streams depending on the requirements.
    - Geo streams are just like any other GS stream and therefore can have events modeled on them, can have rollup virtual streams and can be used in stream derivation calculations (i.e. calculate the distance between two mobile components).
    - There are three ways to set your component's current location:
        1. Manually enter Lon, Lat and Alt.
        2. From an address. Enter the address and click the **Set Coordinates from Address** button
        3. Click on the component's location within the map to the right. Zoom the map out, pan the map and zoom back in to place your component where it is located.

Component Streams Overview

A Stream represents a collection of one or more data points. Each data point can be associated with a single time stamp or a time range. Datetimes are accurate to the millisecond.

All streams must belong to a component.

It is not required for a component to upload all of its streams at the same time with the exception of mobile components. Mobile components are required to upload their longitude and latitude streams at the same time.

There are three types of streams:

- Regular Streams: Each data point is associated with a datetime. Datetimes can be random. There can be tens of millions of data points for each random stream. Random streams can be converted to interval streams dynamically during event detection, graphing, or derivation.
- Interval Streams: Each data point is associated with a start datetime (inclusive) and an end datetime (exclusive). There can be tens of millions of interval data points for each stream. Interval streams manage data associated with time intervals. Interval streams are commonly used for devices with predetermined sample rates that don't change frequently such as temperature sensors or electric meters. Interval streams can manage interval sizes as small as one second and can track gaps (missing data points) between intervals for data quality analysis. Only interval streams can be derived from stream expressions, but stream expression variables can be interval streams, random streams and point streams.
- Point Streams: There is only one data point. This value can change frequently or infrequently, but only one data point is stored. Think of a point stream as being a random stream that only stores its last value. Point streams are good for holding constant values and for storing stream data where only the last value matters.

Stream Feed Processing Order

There are many stream configuration settings that get applied when a stream's feed is being uploaded. Processing takes place in this order:

1. Stream data is uploaded via a feed, the running of an import, manually entered or is derived
2. Constraints are Applied:
    1. Time Filters are applied to regular and interval streams. Values excluded by the filter are removed (or changed to gaps for interval streams)
    2. Min/Max or floor/ceiling is applied
3. Gap filing is applied for interval streams
4. Virtual Streams are created. The Percent Set to Gap is applied during each rollup calculation
5. Event detection takes place
6. The processed feed is saved into the store

Creating a Stream

Most streams are created from within the component editor. Right click on a component in the Component browser, located on the left side of Observation Studio, and choose **Edit Component**.

Right click on the **Streams** folder and choose **Add – Stream** to add a new stream to the component. Select the type of stream to add (regular, point, or interval).

Stream Order: Streams will appear within the component browser in the order that they are defined in the component model. Drag and drop streams to change their order. GroveStreams will use the stream order to determine which feeds belong to which streams for feeds that do not contain stream IDs or UIDs within the component feed JSON array.

Dynamic stream creation during feed uploads: Streams can also be automatically created when metadata is passed into GS as part of the stream feed PUT. See the batch stream PUT API for examples of dynamic stream creation or download the GroveStreams Ganglia feeder source code for a Java example of dynamic stream creation.

Stream Groups

Streams can be grouped within a component. Right click on **Streams** and choose

Add – Stream Group. Add or remove streams from the group by dragging and dropping them.

Common Stream Fields

All three stream types share a common set of fields. These fields are discussed in this section.

- General Properties
  - Name: The name of the stream. A stream can be renamed at anytime.
  - Description: Optional. The stream's description.
  - ID: Optional. This ID can be included on Feed JSON calls to associate part of a feed with this stream. Stream IDs must be unique within each component. It is highly recommended to use component IDs and stream IDs within Feed API calls. IDs can be changed at any time. GroveStreams will associate an internal id, referred to as a UID, with this stream when it is saved. UIDs are used internally and can be used in Feed PUT calls but are not recommended.

- Data Type: Stream data types can be numbers, dates and times, text, booleans (yes, no), longitude, latitude or elevation. Select the data type for the stream.
  - Data types can be changed later and will trigger a component reconcile to occur when the component is saved. It is not recommended to change the stream data type frequently as it may require a lot of processing on the servers to convert existing stream data if there are a lot of existing samples in the store.
  - Select a type with a small value of bytes to keep storage costs low. For example, it requires half as much storage to store float (4 bytes) instead of double (8 bytes). Select float if the precision of doubles is not needed.
  - Text data types are currently restricted to 200 characters per sample.
  - GroveStreams will attempt to convert JSON feed values to the stream type. Values that cannot be converted will be:
    - ignored for regular streams
    - considered gaps for interval streams
    - set to null for point streams
- Unit: Optional. Select a unit for the stream. Units are also used to format numbers and booleans when displaying them.

- Constraints
  - No Constraints: Indicates that there is no constraint.
  - Has Floor and Ceiling: Applies to numerical streams only. If a stream value is greater than the ceiling, then the value will be changed to the ceiling amount. If a value is less than the floor, then the value is changed to the floor.
  - Has Minimum and Maximum: Applies to numerical streams only. If a value is greater than the maximum, it will be set to a gap (or null value). If a value is less than the minimum, then it will be set to a gap (or null value).
  - Is Counter Stream: Applies to numerical streams only. Adds each new sample to the previous sample. Only new samples are summed to the head of a stream. Historical sample changes are not combined with the previous sample. Deletes will not change any existing values outside the delete range. Counter streams are not highly scalable. A component is locked during the inserting of intervals. Frequent updates by many sources to a single Counter stream will not scale well. Use a different technique to aggregate many streams.

        - Reset Cycle: How often the counter stream should be reset
        - ResetValue: The value to set to the first sample that arrives within a new cycle.
- Visuals
    - Default Chart Type: Optional. Applies to numerical streams only. The default chart to display for the stream.
    - Multiplier: Applies to numerical streams only. The multiplier is only applied prior to viewing or graphing a stream.
    - Offset: Applies to numerical streams only. The offset is added to a value prior to viewing or graphing the value. The offset is applied prior to the multiplier.
    - Has Chart Minimum and Maximum: Applies to numerical streams only.

Regular Stream Fields

- Rollup Calendar: Optional. Select a rollup calendar if access to rollup virtual streams for this stream is desired. Selecting a rollup calendar will cause many virtual streams, one for each rollup cycle and function, to be created and stored while data arrives within the same store transaction. This ensures the rollup values are consistent with base cycle interval values and provides extreme performance benefits for streams with small sample sizes as the rollup will not need to be calculated as it is requested. It only needs to be queried from the store.
- Default Rollup Method: Used when a default virtual stream is needed. This usually occurs during derivation, aggregation, or graphing. For example, for a Temperature stream, it would not make sense to choose SUM for the default rollup method. MAX, AVG, or MIN probably make more sense for a Temperature stream. Where a commodity stream such as Water Usage Cost, would probably have a default rollup method of SUM.
- Base Cycle: Select the base cycle for this stream. The base cycle is the first cycle that samples will rollup into. It does not have to be a cycle that exists within the selected Rollup Calendar but it must fit evenly into one of the Rollup Calendar cycles.
- Delete Profile: Optional. Select which sample time range to delete daily by configuring and selecting a Delete Profile. It is highly recommended that all regular and interval streams reference a delete profile even if all samples are to be retained. Someday a decision may be made to remove old samples and it will be much easier to edit a single Delete Profile than it will be to edit every stream.
- Constraints

- Time Filter: Optional. Select a time filter to only include samples for certain times during certain days of the week during certain seasons. Time filters are applied prior to feeds being saved. Time filters are a great way to extract things like:
  - Maximum Weekend Temperatures
  - Average Winter Temperatures
  - Time of Use Energy Charges such as weekend winter off peak usage
  - Whether a door was opened on a weekend afternoon

Interval Stream Fields

- Rollup Calendar: Optional. Select a rollup calendar if access to rollup virtual streams for this stream is desired. Selecting a rollup calendar will cause many virtual streams, one for each rollup cycle and function, to be created and stored while data arrives within the same store transaction. This ensures the rollup values are consistent with base cycle interval values and provides extreme performance benefits for streams with small sample sizes as the rollup will not need to be calculated as it is requested. It only needs to be queried from the store.
- Default Rollup Method: Used when a default virtual stream is needed. This usually occurs during derivation, aggregation or graphing. For example, for a Temperature stream, it would not make sense to choose SUM for the default rollup method. MAX, AVG, or MIN probably make more sense for a Temperature stream. Where a commodity stream such as Water Usage Cost, would probably have a default rollup method of SUM.
- Base Cycle: Select the base cycle, or sample frequency, of this interval stream.
- Delete Profile: Optional. Select which sample time range to delete daily by configuring and selecting a Delete Profile. It is highly recommended that all regular and interval streams reference a delete profile even if all samples are to be retained. Someday a decision may be made to remove old samples and it will be much easier to edit a single Delete Profile than it will be to edit every stream.
- Constraints
  - Percent of Gap Intervals Allowed During Rollup: Use this setting to set a virtual stream's rollup interval to a gap if the number of gaps exceed this amount in the previous rollup. Let's do an example:

        - A rollup calendar is selected to roll hours into days.
        - The Percent of Gaps is set to allow for 50% gaps.
        - If the number of hours that are gaps for a particular day is 13, then all of the hour virtual streams (min, max, sum ,...) will all be gaps (or nulls) for this day.
        - There is some intelligence built into this calculation to avoid rollups appearing as gaps if the cycle being calculated hasn't had all of its intervals uploaded yet. For the above example, the calculation takes place from the datetime of the start of the rollup interval being calculated to the datetime of the last interval uploaded or derived within the rollup interval. This avoids a rollup cycle such as 'day' appearing as a gap until at least 12 hours have been uploaded.

- Gap Filling
    - Gap filling is applied during the appending of stream data. Gap filling is ignored if existing stream data is being updated.
    - Some sensors, such as the GroveStreams Ganglia feeder, rely on Gap Filling. They only upload stream values if the value has changed within a certain time period to reduce network bandwidth.
    - Select the Gap Filling type to be applied.
    - Select the Maximum Continuous Gaps to Fill. Gap filling will not occur if the number of gaps exceeds this amount. The intervals will remain gaps.

File Streams

Interval and Regular streams can store files of any type. Regular streams are the preferred streams for files. Edit a component, create a stream, and set its value type to FILE to enable file storage.

File Streams cannot be derived, but they can be used in derivation. The file URLs will be the variable used in derivation.

Files can be uploaded in Observation Studio by browsing for them or dragging and dropping them onto the file URL grid. Files can be viewed in Observation Studio if the browser supports the viewing of the type file stored. Sometimes, such as with .pdf files, a browser add-in may need to be installed to view a file in Observation Studio.

Leave the time off of the URL to request the latest file for downloading or viewing.

File streams are secured. Add an api_key to the file URL if the file is requested outside a GroveStreams secured session.

Files are not included in Organization Backups today.

Files are GET and PUT using the feed/file api. The GET feed api will retrieve lists of file URLS.

File URLs will be changed internally if a Branded (white-labeled) GroveStreams organization's domain is changed.

Stream Derivation

Stream feeds can be derived instead of being uploaded directly.

There are three types of stream derivation: From RSS feeds, from expressions and from aggregation. Each is discussed next.

Streams can also be derived from file imports, but imports are not discussed within this modeling guide.

From RSS Feeds

All three stream types support derivation from RSS feeds.

The Internet contains thousands, if not millions, of available RSS feeds. RSS feeds are a great way to import data into streams so that they can be used in derivation calculations, be graphed or have events placed on them.

To add a new Stream within the component editor, click on the **Derivation** tab and choose **From RSS Feed** to create an RSS derived stream.

Properties:

- Stream data derived from: Enter the URL for the RSS feed.
- Extract from feed: Select the part of the RSS feed body to be extracted.
- Advanced: Optional. Most feed results are formatted human readable sentences which include a value that needs to be extracted such as a temperature, stock quote or currency exchange rate value. Click **Advanced** to parse out the value from the RSS result.
    - Filters: Create filters to include or exclude parts of a feed.
    - Regex Operations: Regex operations are applied after filter. Regex is a powerful tool for searching and replacing elements of text. It is commonly used for RSS parsing. Google Regex to learn more about Regex usage.
- Test: Click the **Test** button to test the current settings.

GroveStreams RSS derivation will run when the component model is saved and then automatically once an hour.

For more complex RSS feed manipulation, use external tools such as Yahoo Pipes which is free. A Yahoo pipe result can be represented by an RSS feed URL which can then be used within GroveStreams for RSS stream derivation or from within the GS dashboard RSS widget.

Create a GroveStreams Sandbox organization for examples of RSS derived streams.

From Expressions

All three stream types can be derived from expressions.

Add a new Stream within the component editor, click on the **Derivation** tab and choose **Expression** to create a stream derived from an expression.

A Derived stream is calculated from an expression that can include any other component streams within an organization. Calculations occur automatically as dependent stream data arrives. The derived stream will calculate out to the date of the latest dependent's samples. Different stream types, data types, roll-up cycles and interval offsets can be used inside a derivation expression.

Derived streams are just like other streams except that data is derived by the GroveStreams' derivation engine. They can be graphed, monitored and used in other derived stream calculations.

Expression variables are streams. If a variable is an interval stream, then the interval offset can be selected for that variable. For example, if an hourly interval with a time span of 2:00 pm to 3:00 pm is being calculated, a dependent variables' offset can be set to -1 and that variables' data for time span 1:00 pm to 2:00 pm will be used in the calculation. Offsets are useful for things like calculating rolling averages or comparing monthly costs.

Derivation will derive out to the latest sample time of the dependents. Calculations will still occur if the latest time(s) are into the future.

Delete all of a derived stream's samples to recalculate an entire Derived stream.

Expressions can contain any stream data types, conditional "if" operators (<=,>=, <,>, !=, ==, &&, ||) and many functions (i.e., trig functions, log functions, exponents, absolute value, random number and many more).

Click the **Derivation Functions and Details** button within the component editor stream derivation tab to see a more detailed list of expression capabilities.

If any dependent interval is a Gap, then the derived interval will be a Gap.

Example of a derived stream expression that calculates one second 3 point rolling averages from component1.stream1:
Variables:

Variable	Offset	Stream	Cycle	Cycle Function
n_minus_0	0	component1.stream1	Second	-
n_minus_1	-1	component1.stream1	Second	-
n_minus_2	-2	component1.stream1	Second	-

Expression:
(n + n_minus_1 + n_minus_2) / 3

Other usages of Derived Streams:

- A Stream that filters out the Winter on Peak kilowatt usage and applies block charges
- A Stream that converts Fahrenheit to Celsius
- A Stream that calculates average summer weekend afternoon peak temperatures across several cities
- A Stream that sums all the energy for all plant meters (aggregates a collection of component streams)
- A Stream that calculates energy cost every minute, hour, day, month and year
- A Stream that measures the distance between a component and another component every second
- A Stream that applies a diversity factor or multiplier

Dependent streams can have different data types. A stream defined as text can be derived from the sum of two other streams of types long and double. GS will attempt to do the data type conversion wherever it is possible. If the conversion is not possible, the resulting interval will be considered a Gap.

Some intervals may be recalculated every few seconds until they have been 'closed'. For example, if a derived interval stream has a base cycle “Daily” and it is derived from several “Hourly” cycle streams that upload their feeds hourly, the “Daily” interval will remain 'open' for the entire day and recalculated every hour as the dependent's data arrives. A “Daily” interval will be marked as 'closed' and will not be calculated again once all of the dependent's samples have arrived for a “Daily” interval and the “Daily” interval has been calculated.

Derivation is an appending process. Changing dependent's data for intervals that are 'closed' will trigger, by default, a recalculation of 'Closed', or historical, intervals. Right click on the stream within the component explorer and choose **Delete All Regular and Interval Stream Data** to force a total recalculation of the derived stream.

The maximum number of derivation variables is 15. Consider using more than one derived stream or using stream aggregation if the calculation requires more than 15 stream variables.

Only dependent stream samples that have matching datetimes will be used to calculate a sample for derived regular and point streams. An interval stream's end date will be used as the sample time during derivation for dependent streams that are interval streams.

Any errors that occur during stream derivation will arrive as Job Notification errors. This is because stream derivation is a system job that runs every few seconds.

Create a GroveStreams Sandbox organization for an example of an expression derived stream.

Derivation Effective Date: Enter a start date and time when calculations should begin.

Advanced:

- Perform derivation when any variable is null: This will allow a calculation to take place even if one of the dependent values are null. This option typically will force the use of the isNull(..) function within derivation expressions since some values might be null. The result will always be null (a gap) if this option is not enabled and any dependent values are null.
- Perform derivation when:
    - All dependent values have arrived: Derivation will not take place until all dependent streams, used in the expression, have samples that arrived after the last derivation.
    - Any dependent values have arrived: Derivation will take place if at least one dependent stream sample arrives since the last derivation. **Perform derivation when any variable is null** must be enabled with this option.
- Recalculate on dependent historical changes: All derived samples from the date of the dependent change will be automatically deleted and recalculated automatically if any of the dependents (or their dependents) have a sample inserted, edited, or deleted earlier with a time earlier than the tail of the stream (the last sample time).
- Recalculate on any derivation changes: The entire stream will be re-derived whenever any of the stream derivation settings have changed.

- Treat nulls as zero: This will treat any null values (gaps for interval streams) as zero in the evaluation.
- Dependent changes since last run: This is a read-only value that indicates how many dependent streams have changed since the last run. Derivation will not run unless this number is greater than zero whenever there are dependent streams.

Expression Variables: Expression variables are streams that are part of the expression.

- Click the **Add** button to select other streams that are part of the expression. Select a row and click the **Remove** button to remove a stream. Click **Pin for Drag and Drop** to drag and drop sibling streams into the Expression variables table. Don't forget to un-pin the **Pin for Drag and Drop** button after all sibling streams have been selected.
- Variable Name: Give each variable a name. The name must be unique within the table. Variable names are used within the expression editor below.
- Intvl Offset: Enter an interval offset. -1 indicates to use the previous interval. 0 indicates to use the current interval. 1 indicates to use the next interval (which may not exist and be considered a gap if the current interval is the last one in the store).
- Cycle and Function: Select the cycle and function if the variable is an interval stream or regular stream (virtual streams).

Expression: This field contains the interval or sample expression that is used for the calculation of each interval.

From Aggregation

Only interval streams can be derived from aggregation.

Add a new Stream within the component editor, click on the **Derivation** tab and choose **Stream Aggregation** to create a stream derived from aggregating a set of other streams.

Streams can be derived from aggregating anywhere from two to thousands of streams. Calculate statistics such as the maximums, minimums, sums, averages and gap counts for each interval in the aggregation time range. Aggregation is started:

1. Manually by right clicking on a stream within the component browser and choosing **Run Stream Aggregation Now**.
2. Automatically via a Runnable. Create a Runnable and select a cycle schedule for the runnable.

Aggregate last: Choose the time range to perform aggregation by selecting how far in the past to do the calculations. The range is calculated back from the current date and time. If using a Runnable schedule

to automatically run aggregation, then try and select a range that is slightly larger than the schedule. For example, if the Runnable schedule is 6 hours then select an 'Aggregate last' range of 8 hours to ensure aggregation captures new streams or streams that have uploaded their feeds later than the prior aggregation range. It is OK to overlap the aggregation time range as each interval result is recalculated.

Stream Group: Select the stream group to aggregate.

Stream Group Aggregation Function: Select the aggregation function.

Percent of Gap Intervals Allowed During Aggregation(0-100%): Enter the percentage of gaps allowed for each interval calculation. The result will be set to a gap if the percentage exceeds the entered amount.

Optional stream group virtual stream cycle and functions: Optional. Select a cycle and function to aggregate virtual streams.

Results from last aggregation: Displays some statistics from the last aggregation run. These statistics are helpful for confirming the number of streams and intervals that were used for the calculation.

Component Event and Notification Modeling

A collection of Events can be modeled for each component.

Events are associated with streams, and when triggered, have a start datetime, an end datetime, and a location (where the component was when the event triggered).

Starting and ending action packages can be associated with an event. Each action package defines a set of tasks that are executed when the event starts and/or ends. Tasks can be defined to send emails, SMS notifications, GS notifications or make HTTP calls.

Active events can be viewed or canceled within the studios. Event icons will appear within dashboard maps at the location the event occurred at for all active events.

Latency Event

A latency event is used to detect when a stream's feed has not been appended (updated for point streams) within a certain amount of time. The trigger will end when the next update is made.

Right click on the **Events** folder within the component model editor and select **Add – Event – Latency Event** to add a latency event.

Fields:

- Name: The name of the event.
- Apply to Stream: The stream the event applies to.

- Event Category Type: The event category.
- Icon: The icon that is used to represent the event.
- Start Action Delivery Frequency(seconds): In seconds. How frequently to execute the start action package tasks after the event has been triggered. Zero will indicate to only execute the tasks once.
- Start Trigger Action Package: Optional. A set of tasks to execute when the event is triggered.
- Stop Trigger Action Package: Optional. A set of tasks to execute when the event ends.
- Update latency (seconds): In seconds. The number of seconds allowed to elapse before the event is triggered.

Trigger Event

A trigger event is used to monitor stream values. An expression is applied to each sample as it is uploaded. The event will be started when the expression returns 'true' and will end when a value is uploaded and the expression returns 'false'.

Numerical Data Type Expression Examples:

- value > 0.5
- !value
- value == 0
- value * 10 > 100
- (value > 5 && value < 10) || value > 100
- sin(value) > 0.5
- round(value) > 1

Text Data Type Expression Examples:

- len(trim(value)) > 0
- lower(mid(value, 3, 5)) == 'abc'

Right click on the **Events** folder within the component model editor and select **Add – Event – Trigger Event** to add a component stream latency event.

- Name: The name of the event.
- Apply to Stream: The stream the event applies to.
- Event Category Type: The event category.
- Icon: The icon that is used to represent the event.
- Start Action Delivery Frequency(seconds): In seconds. How frequently to execute the start action package tasks after the event has been triggered. Zero will indicate to only execute the tasks once.
- Start Trigger Action Package: Optional. A set of tasks to execute when the event is triggered.
- Stop Trigger Action Package: Optional. A set of tasks to execute when the event ends.
- Value used in validation expression: Optional. Allows virtual stream values to be monitored.
- Validation Expression: Enter an expression that will be applied to each stream value to determine if the event should start or end. The expression must return a Boolean (true/false) result. A 'true' expression result will cause the event to start. The trigger event expression engine is the same expression engine used for stream derivation and therefore supports all the same operators such as 'if' statements.
- Start Action Delay (seconds): In seconds. This setting is used to not trigger the event until the delay amount of time passes while the event is ongoing. This property is useful for ignoring anomalies or for defining different actions to take as an event continues.
    - The start and end time of events will be the sample times. It will be the interval **end** dates for interval streams.
    - `Example: Event expression: value > 0; 1 hour stream sample rate; 2 hour delay:
        - Time: 1:00 Value: 1
        - Time: 2:00 Value: 1
        - Time: 3:00 Value: 1
        - Time: 4:00 Value: 1 (event started – 1:00 is reported as the start)
        - Time: 5:00 Value: 0 (event ended)
        - An event will be triggered at 4:00 and notifications sent out at 4:00 because a delay of 2 hours was set. It was not reported at 3:00 since 3:00 minus 1:00 equals 2 hours which is

not greater than the delay of 2 hours. The notification will report the start time of the event as 1:00 since that's when it really started.

- Dwell (seconds): In Seconds. The amount of time that must elapse between the start of one event to allow the next event to start. This property is useful for avoiding event storms that may send many notifications in a short period of time. Zero indicates not to dwell at all.
    - The time used to determine when an event started for dwelling will be the sample times for regular streams and the interval **end** dates for interval streams. The time between the two must be greater than the dwell amount for the event to trigger.
    - Dwell may be used in combination of a delay.
    - Example: Event expression: value > 0; 1 hour regular stream sample rate; 2 hour dwell:
        - Time: 1:00 Value: 0
        - Time: 2:00 Value: 1 (event start)
        - Time: 3:00 Value: 0 (event stop)
        - Time: 4:00 Value: 1 (dwelling)
        - Time: 5:00 Value: 1 (event start – 5:00 is reported as the start)
        - Time: 6:00 Value: 1
        - Time: 7:00 Value: 0 (event stop)
        - An event started at 2:00 and ended at 3:00. A second event condition was detected at 4:00, but was not triggered due to the 2 hour dwell time because 4:00 minus 2:00 equals 2 hours. The elapsed time must be greater than the dwell amount which is 2. The 5:00 condition was detected and triggered since the dwell time had elapsed. The 5:00 event notifications were sent out when the 5:00 sample arrived. The notification will report the event start time of 5:00.

Complex Stream Event Modeling

Although trigger events support complex expressions, they do not directly support event monitoring that involve previous intervals or other stream values. Use a derived stream and have a trigger event monitor the derived stream values for these types of complex events.

This design is intentional. Event processing occurs as data arrives, within the same transaction, so it needs to be fast to keep feed upload times fast and scalable. Derived stream processing is designed for complex, potentially time consuming, analytics and are a great tool to help model a complex event. The derived stream will keep a record of the expression results for each interval.

How to model a complex event with a derived stream: There are many ways to do this. One way is to create a derived stream with a data type of Boolean (true/false) and a derived stream expression that returns true for the condition to start the event. Then setup an event that monitors this stream for true values.

Action Packages (Including Email and SMS notifications)

Action packages contain a set of configured actions. They are associated with stream events and are executed when an event starts or ends.

Action packages are managed from within Component Studio.

Action Package Actions:

- Email: Sends emails to a collection of recipients.
- SMS/MMS text messages. Sends SMS and MMS text messages to designated recipients. Attach a media URL to MMS to include an image, audio, video and other formats embedded in the message.
- GroveStreams User Notification. Sends GroveStreams user notifications to selected recipients.
- HTTP Calls. Makes HTTP calls.

Action details are template based so that only a few action packages are required for an entire organization. Any part of a message that is surrounded by brackets '[]' will be replaced during the action execution. Click on the **Message Variables** button to see a list of variables that can be embedded within an action details.

Reconciling Component Changes

A component model, including stream changes, is allowed to change even if data already exists and feeds are being uploaded.

For example, when a stream's base cycle interval size or data type is changed and the stream has several years' worth of one second data, a Reconcile job is kicked off and run asynchronously. Feeds can still be uploaded/downloaded while the job is running, but the reconcile job requires that a feed only append data while the job is running.

Cycles, Rollup Calendars and Virtual Streams

Cycles and Rollup Calendars can be associated with interval streams to provide virtual stream statistics as data arrives. Associating a rollup calendar with a stream can create as many as 80 virtual streams.

Virtual streams are streams of data calculated from rolling-up or aggregating the base cycle stream intervals into larger time intervals (i.e. rolling up hours into days. The Daily Min, Daily Max, Daily Sum would each be a virtual stream).

Virtual streams are an important part of GroveStreams as they allow for:

- Zooming in and out of time ranges
- Extracting statistics over time periods such as hours, days, and months.
- Graphing rollup statistics as base data arrives
- Using rollup statistics for derivation calculations
- Using rollup statistics for event detection

Rollups require a time zone id so that the start and end times of each cycle are known. The time zone id can be obtained from the component, the cycle or sometimes it is gotten on the fly from the user's time zone or from the browser time zone setting depending on what action is taking place.

Cycles

A cycle is a defined set of time intervals without any time gaps between intervals.

Cycles are used to:

- define the size of intervals for interval streams
- define rollup calendar levels
- schedule how often runnables automatically run

A cycle can be defined to be Fixed or Custom:

- Fixed: A start datetime and a time zone is selected along with "X" number of Seconds, Minutes, Hours, Days, Weeks, Months or Years
- Custom: A collection of start and end datetimes and a time zone are selected

Examples of Fixed Cycles:

- Second Cycle: 1 Second cycles, starting at December 31, 2011 12:00:00 am

- Quarterly Cycle: 3 Month cycles, starting at January 15, 2012 12:00:00 pm

Example of Custom Cycle:

- Seasonal Cycle: Spring, Summer, Fall, Winter seasons

The time zone is used for the start datetime for fixed cycles and for the start and end datetimes for custom cycles. The time zone can be set to use the component's time zone so that the same cycle can be used across time zones.

Cycle Properties:

- Name: The name of the cycle.
- Description: Optional. The description of the cycle
- ID: Optional. Used by the API
- Time Zone: The time zone for the cycle. It is best practice to select **Use Component's Time zone.** The exception is when you would like all of your rollups that reference this cycle to rollup into a time zone that is different from all of your components, such as a corporate headquarters.
- Cycle Type: Fixed or Custom.
- Reference Interval start date and time: Select a date and time that is at the start of any intervals for this cycle. It can be a time in the past or future. The closer the reference date is to the current date, the faster date and time calculations will be on the GS servers. It is recommended to set this value to a date time to about one or two years into the future. This value can be changed later, but it can only be changed to a date time that lands on another interval start date.

Rollup Calendars

A rollup calendar defines how stream data can be rolled-up for viewing and analysis.

Rollup calendars are optional.

If a stream has a base cycle set for one second intervals, it quickly becomes cumbersome to view and analyze a years' worth of data which is 31,536,000 intervals. For the "float" data type which is 4 bytes, a years' worth of data would be about 120 Megabytes. Instead of downloading 31 million intervals and sifting through them, a user is better off downloading a set of rolled-up stream data (i.e. a virtual stream) at a larger interval size and then drilling in on the interesting cycles. A good example of a graph that demonstrates the power of roll-ups is the Google finance Graph. Unfortunately, unlike GroveStreams, Google hard-coded their roll-up cycles (1d, 5d, 1m, 6m, 1y, 5y).

Also, it may be desired to monitor energy usage in 1 second intervals so as to watch for certain events, but billing is probably based on monthly kW and or kWh. A stream with a rollup calendar would be an

ideal solution. GS provides the ability to view/monitor one second intervals and the ability to view/monitor one hour and one month cycle intervals too, at the same time. GS allows for monitoring one month cycle interval change nearly every second... actually, every 10 seconds since feeds can only upload every 10 seconds from each source :(

A rollup calendar is defined by simply referencing a collection of cycles, described above. When the rollup calendar is saved, it will be sorted and validated that it contains no gaps. A stream's feed can only be uploaded to the base cycle. Any stream non-base cycle information can be used (graphing and such) just like the base cycle.

Example:
Let's define a rollup calendar that rolls one second data into several other cycles, up to a year:

Cycles that were defined earlier and that the rollup calendar will reference:

- 1 Second Cycle
- 5 Minute Cycle
- 1 Hour Cycle
- 1 Day Cycle
- 1 Month Cycle
- 1 Year Cycle

After the rollup calendar is defined and associated with a stream, the stream's data for each rolled-up cycle is available as base cycle data is uploaded. When the stream uploads 10 intervals of data (1 second each), all of the rollup data is immediately available via the API and browser user interface. Rollup cycle data can be requested with these functions:

- First : Returns the first interval value of the base cycle
- Last : Returns the last interval value of the base cycle
- Min : Returns the smallest interval value of the base cycle
- Max : Returns the largest interval value of the base cycle
- Avg : Returns the time weighted average of the base cycle
- Sum : Returns the sum of the intervals of the base cycle
- Min Occurrence : Returns the start datetime the minimum occurred for the base cycle, accurate to the second
- Max Occurrence : Returns the start datetime the maximum occurred for the base cycle, accurate to the second

- GapCount : Returns the number of base cycle interval gaps that occurred for this cycle's range. For example, if the base cycle is 1 Second, the requested rollup cycle is 1 Year, and the request datetime range is Jan 1, 2010 12:00 am to Jan 1, 2011 12:00 am, then a single value representing how many 1 second intervals are Gaps is returned.
- NonGapCount : Like GapCount, but returns the number of base cycle intervals that are not gaps for the requested datetime range.
- IntvlCount : Returns the number of base cycle intervals whether they are a gap or not
- MilliSecCount : Returns the number of milliseconds for the requested datetime range
- NonGapMilliSecCount : Returns the number of nonGap milliseconds for the requested datetime range

Gap intervals are ignored for some of the above functions (First, Last, Min, Max, Avg, Min Occurrence, Max Occurrence).

Time filters are taken into consideration for all calculations. Intervals that are excluded by a filter are ignored.

Only base cycle information can be uploaded via interval stream feeds. The other virtual streams are calculated and therefore cannot be uploaded. Virtual streams reflect any changes to base cycle data during any time period. They are all updated at the same time within the same transaction.

Virtual stream calculations are based on base cycle values. For example, for a three cycle rollup calendar minute, hour, day, the calculation for daily Average is not the average of each hour cycle, but is the average of all minute base cycles. This applies to all virtual streams including gap counts which returns the number of base cycle gaps for each virtual stream rollup cycle.

Note that an interval stream's base Cycle does not have to exist within the rollup calendar it is referencing. It just needs to be able to "fit" evenly into one of the cycles and it will be rolled-up from the cycle that it fits into and above. For the example above, a stream might have a base cycle of "30 Minute". It would upload a value for each 30 minute interval and that value would start rolling up to 1 Hour then to 1 Day and so on up to a Year.

What does "fit" evenly mean? A cycle defined as "7 Seconds" will not fit evenly into a rollup cycle defined as "8 Seconds" or "1 Minute" but will fit evenly into a cycle defined as "42 Seconds".

Time Filters

Time filters are applied to stream feeds as they are uploaded or derived. Sample times or interval ranges not included in the filter are converted to Gaps (or null values) for interval streams and ignored for regular streams. This is advantageous for several reasons:

- GroveStreams supports data scarcity when it comes to storage. Most null values are not stored so you'll be charged less for storage.
- It improves performance, since most Gaps are not stored, they do not need to be retrieved
- Gap values created by a Time Filter are ignored by the GroveStreams' Rollup-Calendar engine. For example, the request for a Cycle Gap Count will ignore intervals excluded by the Time Filter.

Time filter, in combination with rollup calendars, are useful for extracting statistics such as:

- Energy time of use billing determinants, such as winter off peak energy
- Determining if a door was opened within a certain time range
- Determining seasonal daily averages

Time Filter Properties:

- Name: The name of the time filter
- Description: Optional. The description of the time filter
- ID: Optional. Used by the API
- Time Ranges (Seasons):
    - Time Zone: The time zone to be used for the Time Filter. Best practice is to choose **Use Component's Time zone**.
    - Time Ranges: Enter each time range to include for this time filter.
        - ***Important***: Entering no ranges indicates to include any time range.
- Time of Day
    - Minutes Per Interval: Select the number of intervals to be used for filtering. **Double click** on an interval to include/exclude it or select it and click the **Include** or **Exclude** buttons.

Regular streams and time filters:

Each sample's datetime is used to determine if it is to be included. The time filter's start datetime is inclusive and the filter's end datetime is exclusive.

Interval streams and time filters:

Each interval must be totally included within each of the time filter's inclusion ranges to be included. Best practice is to define your time filter with as few intervals per day as is needed. Streams with base cycles smaller than the time filter's intervals are compatible as long as their intervals fit evenly into the time filter intervals. For example, a one minute base cycle stream is compatible with a time filter defined as having 1440 minutes per interval (one day intervals).

Component Auto-Registration

GroveStreams has the capability to automatically create component and streams as feed samples arrive. New components will be placed within the component root folder within Observation studio.

There are several ways to accomplish component stream registration.

Automatic Registration with Defaults

Automatic registration is useful to get up and running quickly. Most users will discover that the default component and streams do not accomplish what they need and will need to move to another registration technique.

GroveStreams will create a default component with default streams as feed PUT calls arrive. The component name will be the ID of the component, the owner of the component will be the organization owner, and the time zone will be the organization owner's time zone.

Default streams will be regular streams. The name of the stream will be its ID. The value type of the stream will be either DOUBLE, BOOLEAN (true/false), FILE, or STRING depending on type of sample uploaded. The Base Cycle, Rollup Calendar, Default Rollup Method, and Delete Profile will be defaulted to the ones set within Admin -Organization - General Settings – New Stream Defaults.

Manual Registration

Manual registration is fine for small users with just a few components.

A new component with streams can be created manually within Observation studio. The feed PUT will place the samples into the component and streams with the IDs used in the feed PUT.

Registration with Component Templates

Registering components with Component Templates is the most common way to register new component and streams when the number of streams and each stream's settings are known in advance.

Large organizations typically have a large number of devices that are identical (such as utility electric meters). It makes no sense to manually create a new component instance by hand every time a device comes on-line.

A component template is used to simplify creating multiple component instances that only differ by a few attributes and stream feed data. When a component uses the feed API to upload data it can pass in a

component template Id. A component instance will be created automatically within the organization, if a component does not already exist for that feed (also known as device automatic registration).

Component Template Linking

A component is "Linked" to the template that created it. Any changes to the template may be applied to all components linked to that template if the organization owner is signed up for a pricing plan that has advanced modeling.

Only changes made to Streams, Events, and Groups (stream folders) are applied to linked components.

Changes can be applied to a live system. Changes will run as a single asynchronous Job.

Removing a Template Link

- Right click on the component and select **Edit Component**
- Select **General Properties** on the left
- Click the **Remove Template Link** to un-link this component from a template. Removing a link prevents any changes to the template from being applied to the component.

Changing a Template Link

- Right click on the component and select **Edit Component**
- Select **General Properties** on the left
- Click the **Remove Template Link** button

Applying Template Changes to all Linked Components

- Enter Component Studio
- Right click on the template and select **Reconciled Linked Components**
- Click **Reconcile Changes**
    - **To Selected Linked Components Below**: This will only reconcile the components selected below. It is highly recommended that template changes are initially applied to only one component so that the changes can be verified before applying them to all components

- **To All Linked Components**: This will reconcile all components linked to this template

Linking Up Components: Each of a component's streams, events, and groups are individually linked to a template's streams, events, and groups when a component is linked to a template. Each component item stores a template item's id internally. During reconciling, each item's template id is matched up with its templates counterpart and reconciled.

Linking is required for any existing components that were referencing a template prior to the template reconciling enhancement (late 2015) or when many components are having their templates changed at one time.

An item, such as a stream will added if it isn't found.

Items will be sorted to match the template sorting.

All item names and properties will match the template.

Components can be extended: Any items, such as streams, can be manually added to a component. Template reconciling will ignore these items since these items do not have an internal template item id associated with them.

A component can be linked to a template by editing a component one at a time (described above) or, within Component Studio, by selecting **Link To ->**

- **Selected Components**
- **All Components Already Linked (Re-does links)**

External Derivation Stream References and Template Reconcile

Select **Reconcile Lock** for a derivation dependent stream on the derivation grid to preserve a component's derivation dependent referenced outside the component during template reconcile. A locked dependent must keep its variable name the same between versions as it's used to determine the row to be locked (not changed during reconcile). The interval offset, cycle, function, and statistic can still be modified for locked stream and those changes will flow through.

Registering new Streams On-The-Fly

Sometimes the number of streams is not known at modeling time and these streams need to be dynamically created and added to components as feeds include them.

By default, GroveStreams will create a regular stream when a Feed PUT arrives with a stream ID that does not exist within a component. The default regular stream will not satisfy many scenarios.

Include the optional dtId and dsId URL parameters as part of the feed PUT call to have all new streams be created based on a component template stream. dtId is the ID of the template and dsId is the ID of the stream.

Advanced Registration

Combining Component Templates with Dynamic Stream Registration

The compTmplId and dtId/dsId parameters can be included at the same time. When this occurs a new component will be created based on the template ID along with all of the streams within the template. Any streams arriving within a Feed PUT that do not exist within the newly created component based on the template, will automatically be added to the component based on the dtId/dsId parameters.

This type of registration allows for the registration of components with a predefined set of streams and also allows for new streams to be appended to the component as they arrive.

Overriding Template Settings within a Feed PUT

The GroveStreams advanced feed PUT API allows for setting certain component and stream attributes after they have been created during a feed PUT. This is accomplished by including a "defaults" section within the feed PUT JSON body. Attributes such as the component's name, description, time zone, location and others can be set during the creation of the component. Stream attributes can also be set within a JSON stream "defaults" section.

Passing "defaults" with every feed call can be a waste of network resources. To avoid passing this metadata with every call:

1. Include a feed PUT URL parameter called "createDefaults" and set it to false. This prevents any missing streams from automatically being created if they are not included within a template and the dtId and dsId parameters are not used.
2. Monitor the feed PUT response JSON body and check for a "missing" JSON array. The missing JSON array will list every stream that was missing in the previous PUT call.
3. Loop over the missing array and include the stream "defaults" section with the stream metadata with the next feed PUT call.

Command and Control

GroveStreams supports controlling and configuring devices from dashboards. The steps are simple:

1. Create a stream for each command or setting variable. Point streams are recommended if a record of commands is not required. Regular streams are recommended if a record is required.
2. Create a dashboard and add the Stream Feed Form widget to it.
3. Drag the control and setting streams onto the widget.
    a. The Stream Feed Form widget will append the values to Regular streams and update the value for Point streams.
4. On your device:
    a. Use the GroveStreams Feed PUT API (all on the URL) and include the IDs of the command and settings streams by adding rsid parameters to the URL. This will force GroveStreams to return the last_value of each rsid stream while uploading samples.
    b. Have your device check the return stream values after each Feed PUT call.

This technique is simple, can keep a record of your commands, and works through a local firewall without having to configure the firewall.

See the Arduino YUN – Control Example for a good example of designing your own command framework.

Blueprints

Blueprints allow the copying of an organization's models to another organization.

Why would blueprints be used? If a company were selling smart plugs, blueprints would prevent their customers from modeling their smart plug environment within GroveStreams. The smart plug company would start with creating an organization that includes all of the modeling for auto-registration of components and the displaying of energy usage and costs within dashboards. Then the company would create a blueprint based on this organization and create smart plug customer's organizations based on the blueprint so that they can be up and running without doing any modeling.

To create a blueprint, enter the organization and choose **Admin – Export – Entire Organization to a blueprint**. The blueprint is downloaded to the local computer as a text file and can be used during the creation of a new organization.

Advanced Blueprint Operations

More blueprint options are available if your pricing plan includes advanced modeling such as:

- Importing a blueprint after an organization has been created
- Selecting specific items for your blueprint such as components, templates, and dashboards

Export Profiles – Advanced Modeling

Export Profiles allow modeled items to be exported to many other organizations or blueprints.

Export Profile Details:

- Choose **Admin – Export – Models Advanced** to create an Export Profile. This action brings up the Import Profile manager window.
- Items can be drag and dropped from the Components and the Dashboard panels onto the export items grid. Drag Components, Dashboards, or Maps onto the grid or drop entire folders. The entire folder and everything below it will be included in the export if a folder was selected.
- Models are overwritten during the import if they have the same ID. It is a best practice to ensure all Items throughout your organization have their ID properties set if items are copied between organizations.
- Model dependents are included during an export. For example, streams, dependent component and streams used in derived streams, units, action packages and all other dependents are exported automatically when exporting a component.
- Component and Content folders will be created if they do not exist within the destination organizations. Each item's Group security settings will be included if the folder or other items did not exist and are being created. Security is left as it is if the repository node already existed.
- Security is enforced during this entire process. The user doing the export/import must have Import and Export capabilities. They must also have the proper rights within the component and content repositories from the source and destination organizations.
- Organization users are not included as part of the export. The following items will have users removed from them during an export:
    - User Groups
    - Action Package Packages
    - Content Security (folders, dashboards, components, maps)

- It is highly recommended to experiment with a couple of test organizations before attempting to use the wizard on a live organization.
- Choose **Admin – Import Models Within a Blueprint** to import a blueprint into an existing organization.

Modeling Best Practices

- Walk through the GroveStreams sandbox guide before doing any modeling. It is a good introduction to many GroveStreams concepts.
- Use Component Templates whenever two or more components are alike.
- Always assign an ID to Streams and Components and use those IDs inside stream feeds. The IDs are also used for advanced modeling to determine if an item already exists when copying items between organizations.
- Model your organization so that components can auto register themselves.
    - Use Component and Stream IDs within stream Feeds.
    - Use a componentTemplateId within stream feeds.
    - Auto registering will make it easier to make changes to the component template and have those changes apply to components who's feeds include the template ID: First make changes to the template and then delete the existing component. The new changes will come into effect when the component auto-registers.
- Model dashboards to use stream groups so that newly registered component streams will automatically appear in existing dashboards.
- Leverage the Runnable scheduler to automatically update stream groups, perform imports and run stream aggregations.
- Try and only have one Rollup calendar so that streams are compatible with each other during derivation or viewing.
- Monitor System notifications while modeling to identify errors quickly.