User Manual

Groundhog Version 147.1

Written by David A. Falk

Legal Information

Overview

The Groundhog Chronology Test Laboratory is a web-based application that calculates and validates chronological theories. The software takes king-lists and synchronisms and from those reconstructs a chronology. This manual will show how to gain access to the application and how to use the application to build your own chronologies.

Login Credentials

To use the system, it is necessary to first obtain a user account. To get login credentials, please send an email to zdfalk@gmail.com with **(1) your first and last names**, a **(2) permanent email address**, a **(3) telephone number**, and a **(4) title for your database**. Once your credentials are approved you will receive an email with the following information:

Username: <lastname+first initials>

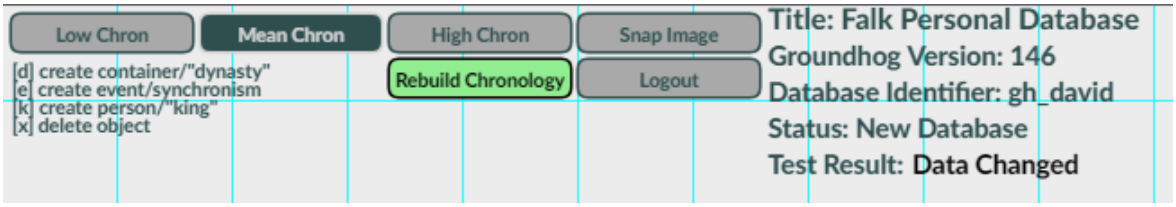
Password: <random strings>

URL with your login page: https://www.groundhogchronology.com/chronology/users/<Username>/index.html

The username will typically be the first six characters of your last name followed by the first initial of your first name plus a numeral. For example, if your name is John Smith, your username will be like smithj1. The URL with your chronology is password protected. And only the account owner can make changes to your data set.

Understanding the User Interface

The user interface uses a simple node system. When you receive your login credentials, you are given access to the application and the backend software which connects to an instance of an unpopulated MySQL database. You will be presented with a blank instance with a status header that looks something like the following.



Header (Middle and Right)

This is status header contains all the basic functions needed to run the software. The header is divided into three sections: the software logo is to the right, five information messages are in the center of the header, and software controls are found to the left.

Title

The top three lines of information messages are static. The top line is the title (in English words) for your database. This was the title for your database that you requested when your account was created.

Groundhog Version

The next line down is the current version of *Groundhog* that was used. This is just information of which version of the software is being used on the front of the system.

Database Identifier

The third line down is an identifier for your database. That identifier will always begins with a gh_ prefix and will most likely be a combination of the prefix with your username (e.g., gh_smithj1).

Last Update

The fourth and fifth line of the information messages will be variable based upon what is happening with the data processing. The fourth line tells you the last update that was made to your data. If a chronology rebuild was successful, it will tell you that "All Jobs Complete," the date/time of completion, and how long the job took to complete.

Test Result

If you change any data, the fifth line (the "Test Result") will change to "Data Changed." If you rebuild the chronology, the Test Result will show the results of the validation testing. If the chronology is consistent, the text will turn a green color with a message stating that the

chronology is consistent, a CV that indicates what percentage of the variations were consistent, and the Runtime that it took to complete the rebuild. If the chronology is not internally consistent, the text will turn red with a message stating that the chronology is inconsistent. In the event of an inconsistent result, none of the permutations tested produced a consistent chronology.

Header (Left) Keyboard Shortcuts

On the left side of the header, there is a small cheat sheet of basic keyboard shortcuts. There are four keyboard shortcuts that are available to use from the main page. To use these shortcuts click on a place on the workspace to indicate where you want to place the object or to activate an object to be affected, then press the keyboard shortcut.

Create a container/“dynasty” (letter-d)

This shortcut creates a container or a “dynasty” object. Container-objects can be created inside of other container-objects or directly on the workspace. Only container-objects and person-objects can be created inside of container-objects

Create an event/synchronism (letter-e)

This shortcut creates an event or “synchronism” object. An event-object can only be created on the workspace.

Create a person/“king” (letter-k)

This shortcut creates a person or “king” object. Person-objects can be created inside of other container objects or directly on the workspace. However, it is highly recommended that person objects only be created inside of a container object.

Delete an object (letter-x)

This shortcut deletes an object. If you are deleting a container object that contains other objects. Inner most objects should be deleted first.

Header (Left) Control Buttons

On the left side of the header, there are five control buttons. These buttons control a variety of views and server commands.

Low Chron[ology], **Mean Chron[ology]**, and **Mean Chron[ology]**

These three buttons that are in a toggle relationship to each other. The default position for this toggle has the **Mean Chron[ology]** button active. When one of these buttons is active, the button turns dark gray with light gray text. The purpose of these buttons shows the results of a chronology. These buttons do not change the data, but switches between various views of the results. The **Mean Chron[ology]** button shows the mean chronology with the error notation.[1] The **Low Chron[ology]** displays the low chronology (where dates are resolved closer to the present) without error notation. And the **High Chron[ology]** displays the high chronology (where dates are resolved further back in the past) without error notation.

Snap Image

This button downloads a png image of the workspace. The resolution of the images is 2800 by 5300 pixels.

Logout

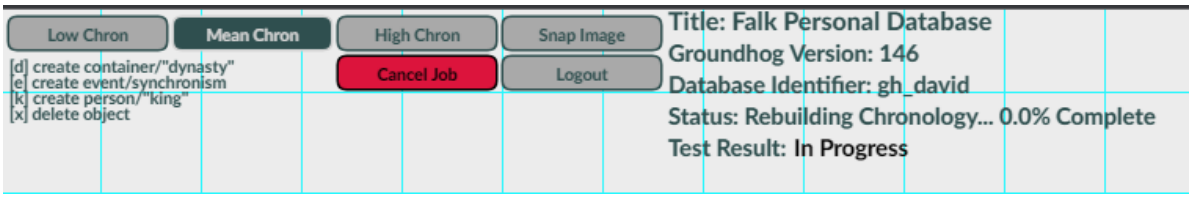
This will log you out of your current session of the client. After clicking on this, you will have to log back into your index page to resume your work with the client. It is recommended that if you are working from a computer system that you do not own to log out after your work to prevent a security breach.

---

[1] *Groundhog* uses (X/Y) notation for error notation, instead of ±X scientific notation. The reason is because sometimes the average occurs during the middle of a year. For example, a king may have ruled 1007.5 BCE ±2.5 years. Using scientific error notation would impractical, since 1007.5 would be rounded up to 1008 anyway, with either plus 2 years or minus 3 years. If it were 1008 (±2) BCE, that would lead to a possibility for the earliest date being 1006 BCE, when 1005 BCE also a possibility. This is why this nomenclature is used instead of normal scientific rounding notation.

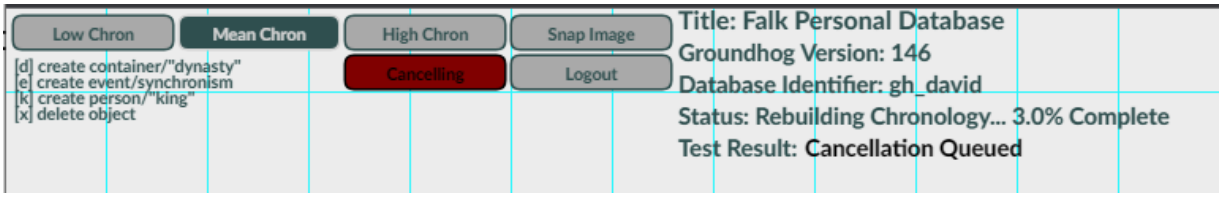
Rebuild Chronology

This button initiates a rebuild of the chronology. When activated, the button will turn red and Test Result will let you know that your job is “In Progress” or that your job has been queued up.



Cancel Job

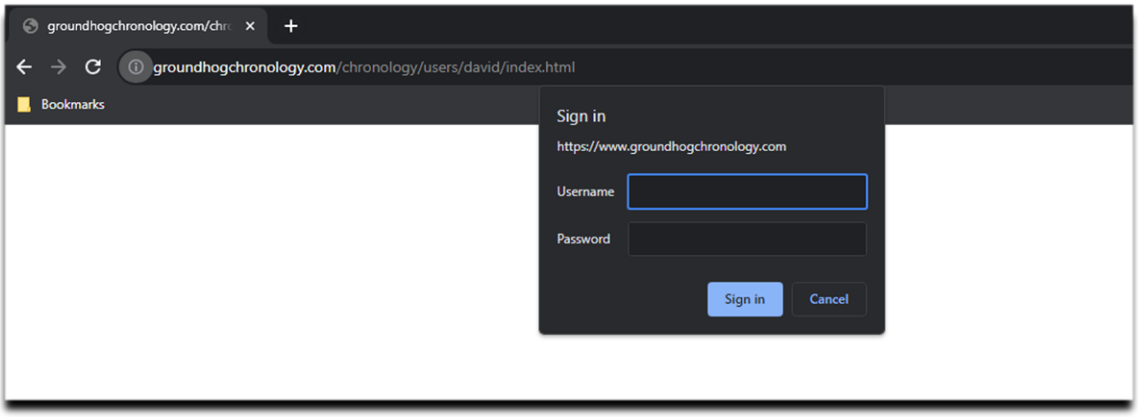
This button initiates a rebuild of the chronology. When activated, the button will turn dark and Test Result will let you know that that the cancellation request is queued.



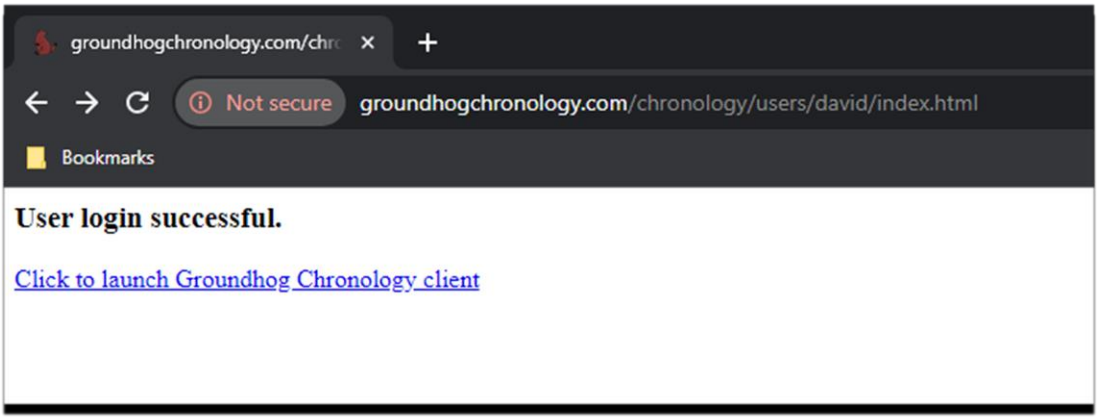
Reference Guide

Loading Your User Interface

After you receive your login credentials and load your index.html URL, the next step is to load the URL into a browser. The software has been written for and tested on the Comodo Dragon web browser, which is freely available for download, but should be capable with other browsers also.

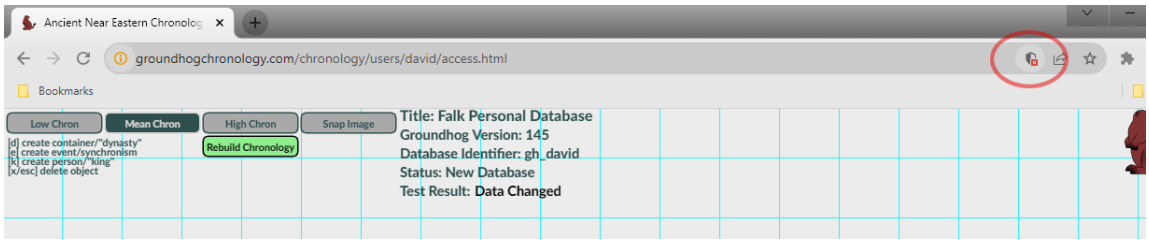


You will be asked for your username and password. If your login is successful, the login page will appear. Click on the link (in blue) to access the application client.

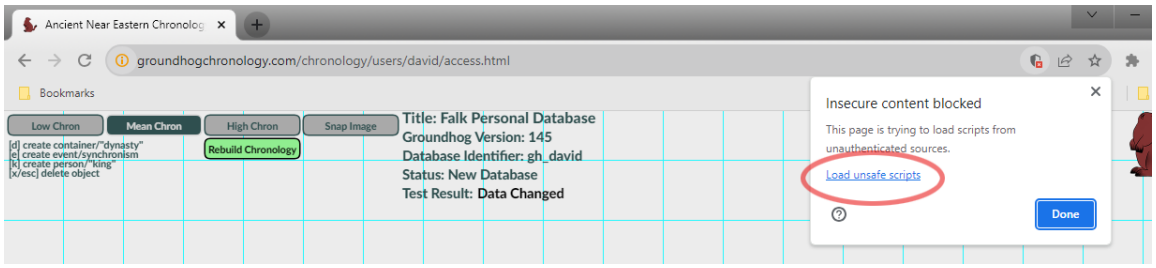


Insecure Content Security Alert

On some systems, you may get the following warning icon by your browser (like the one circled in red). This is a warning by your browser that the client-side JavaScript scripts are not being loaded as a security measure.



Click on the icon, and "Insecure content blocked" window should pop up. Then click on the "Load unsafe scripts" link.



The warning icon should disappear, and at that point, the client software is operational and ready to use.

Container-Objects

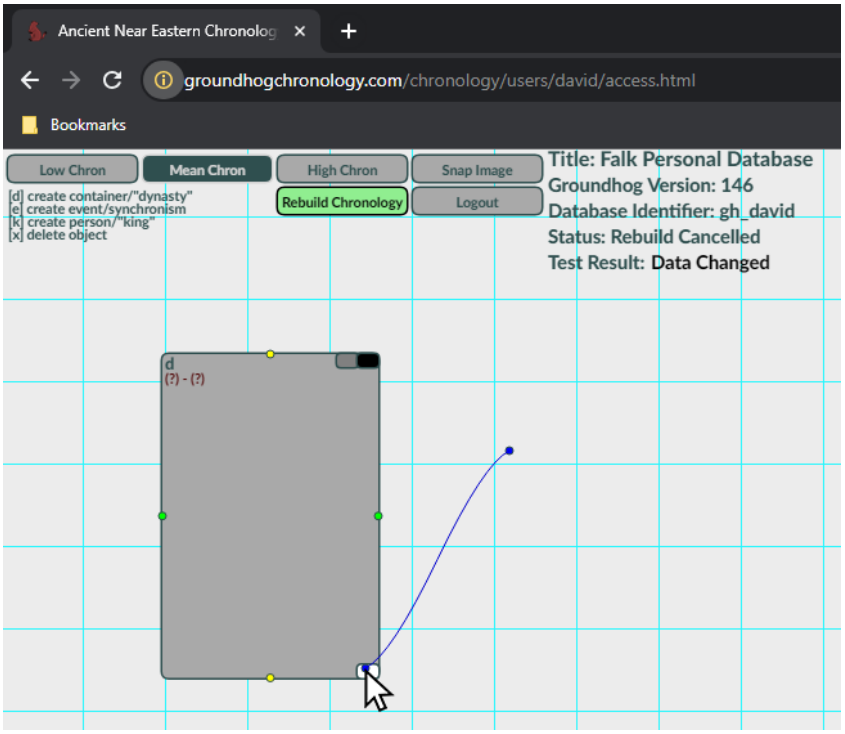
Container-objects are used to encapsulate and manage complex objects. Container-objects can contain other container-objects and person-objects. Software optimizations make the use of these objects necessary because of the large data models involved; however, they add to the convenience for the end-user.

To create a dynasty, you need to move the mouse cursor to where you want to create the first container-object to be created and press the <d-key>. The top-left corner of the container-object will be created under your mouse cursor. The container-object contains two text-objects

that are color-coded black and red; four control-buttons that are color-coded: black, gray, and white; and two node-points color-coded yellow and green.

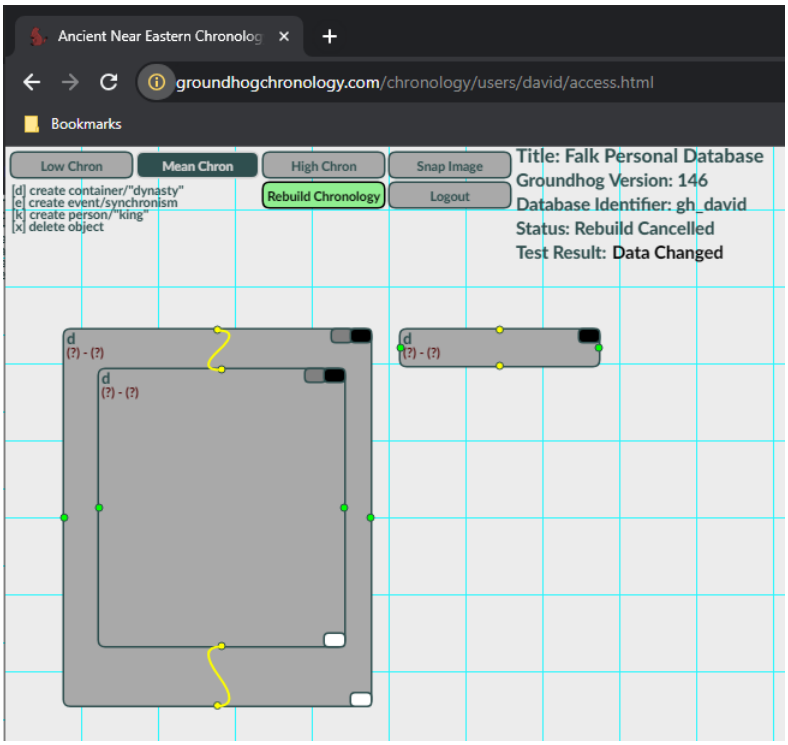
Container-object, "Resize," white button

The white button (bottom right of the object) is used to resize the container-object. To resize a container-object, left-click on the white button and pull to resize.



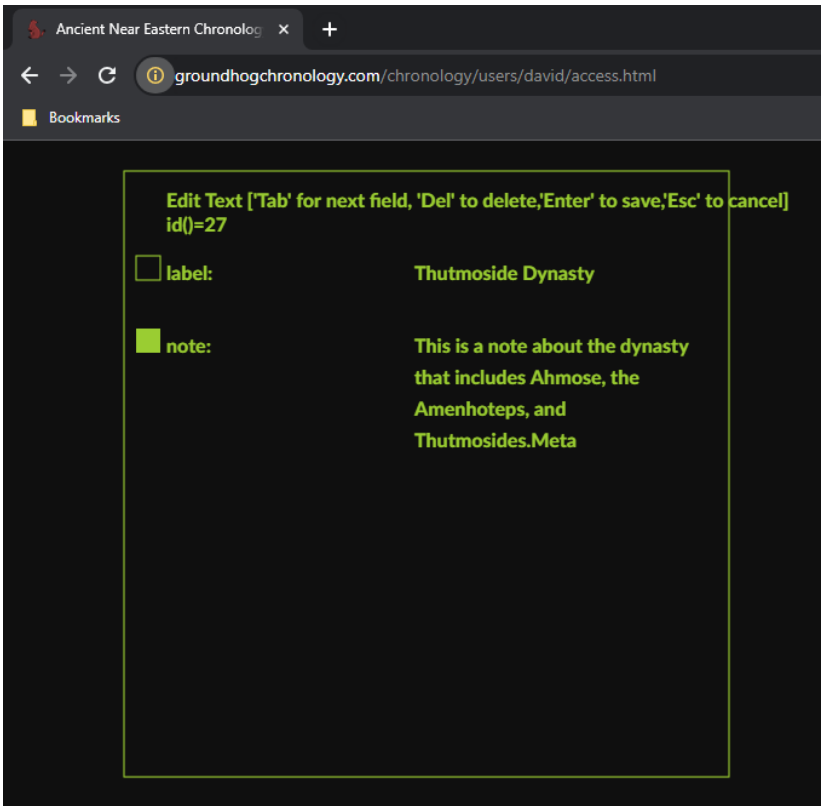
Container-object, “Minimize/Maximize,” black button

The black button (top right of the object) is used to minimize the container-object. Left-clicking on the black bottom will either maximalize or minimalize the object based upon its previous state. When minimalized, the container-object will hide all the objects inside of the container.



Container-object, “Edit,” gray button

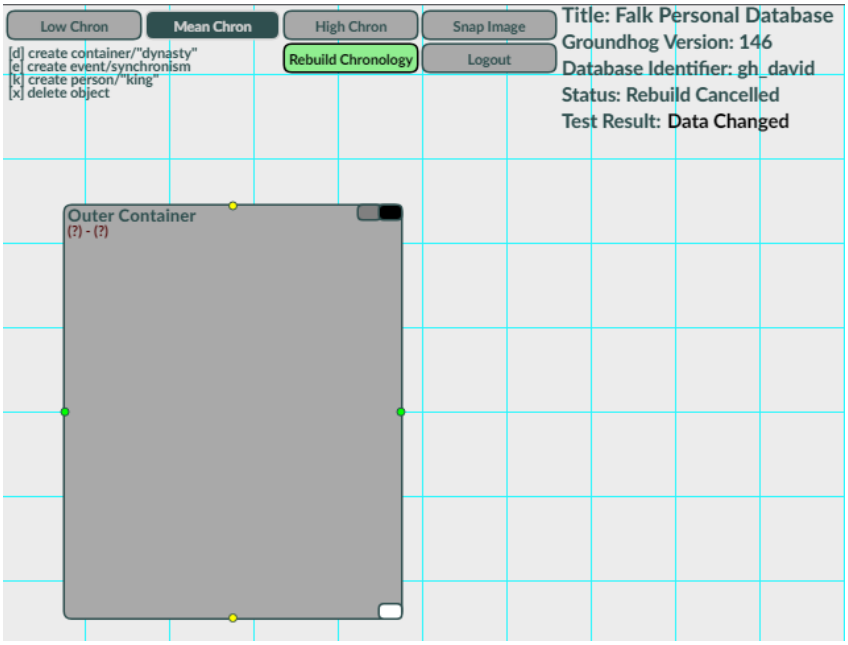
The gray button (top right of the object) is used to edit the text fields of the container-object. When you left-click on the “edit” button, a menu will appear as follows:



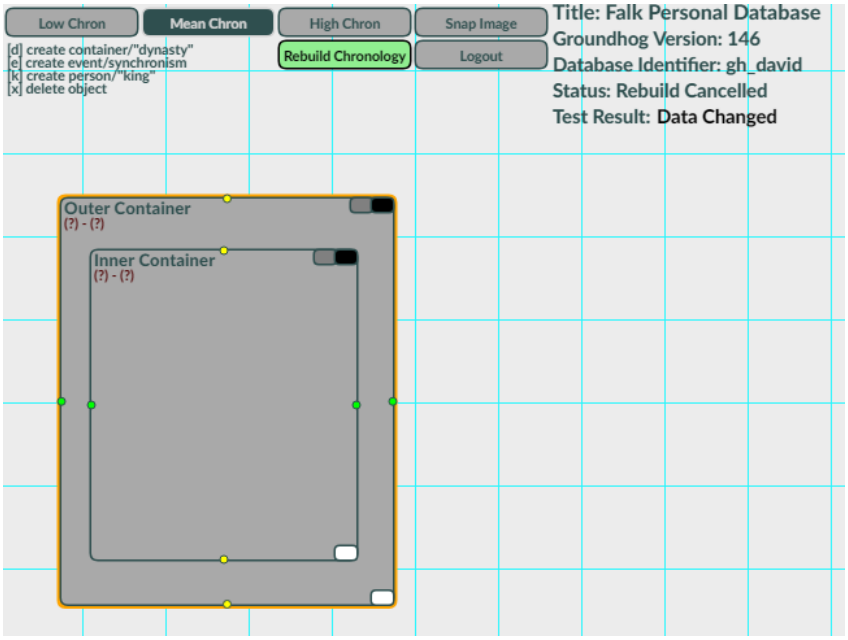
The edit screen shows the fields for the object and the id number of the object. There are two fields that can be edited (label and note). Type <tab> to switch between the fields. The active field is indicated by the white square. Type <del> to backspace what you typed. Type <esc> to exit the menu without saving. Type <enter> to save what you typed.

Nesting Container Objects

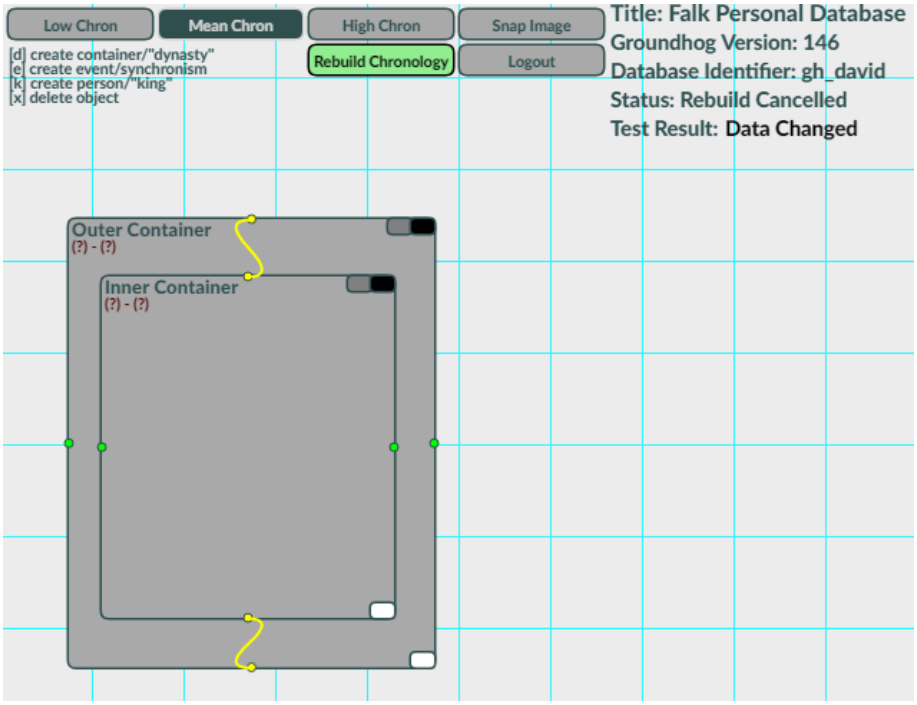
Nesting container-objects is important for constructing chronologies in Groundhog. To nest a container-object, first create a container-object on the workspace.



Then create a container-object inside the first container-object.



Then connect the containers together. Connect the top yellow node-point of the outer container to the top yellow node-point of the inner container. To do that left-click one of the node-points and drag it to the other node-point. Do the same thing with the bottom node-points. The nested containers should look as follows:

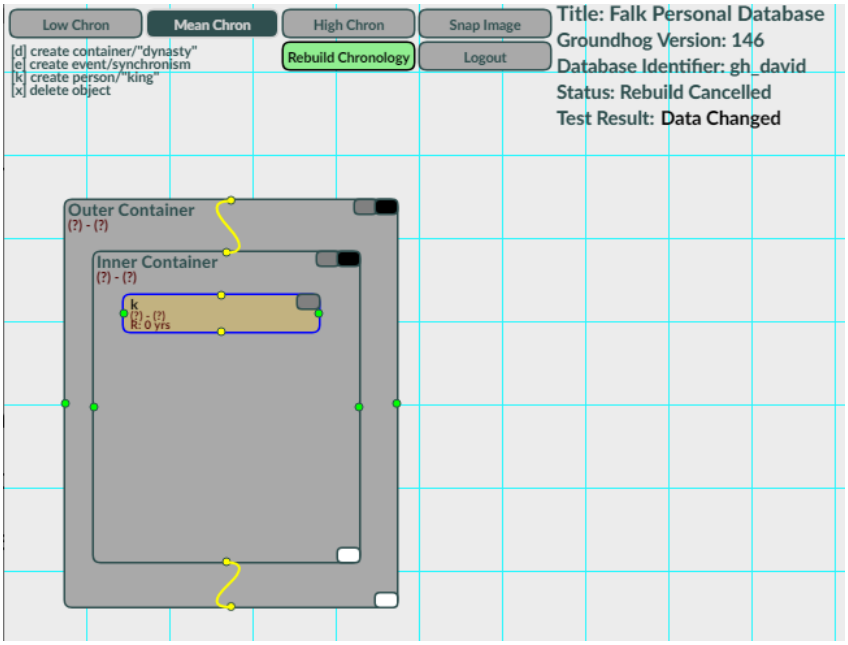


Pro Tip: to unlink an object, right-click on one occupied node and drag it to the other occupied node.

Person-Objects

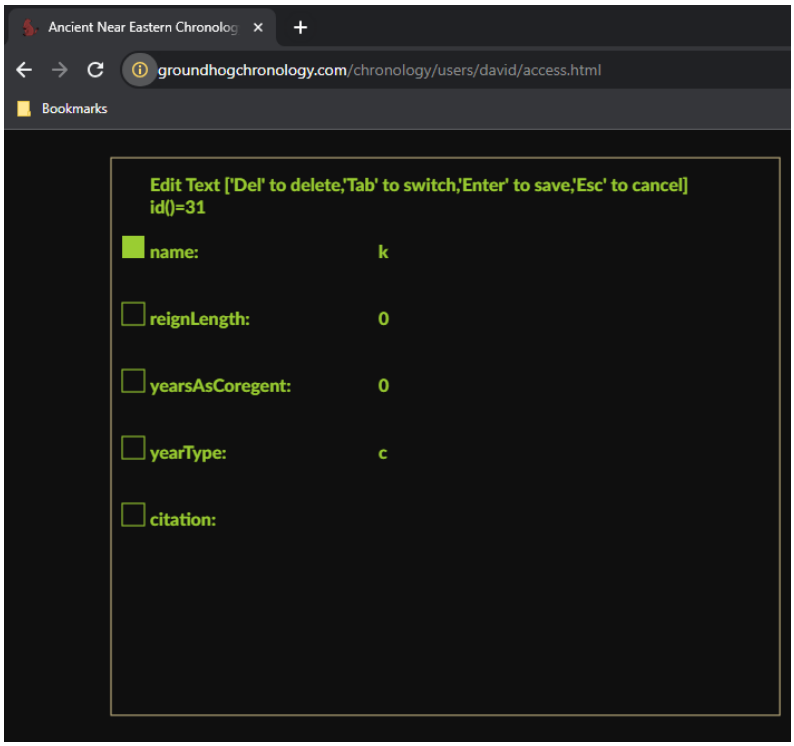
Person-objects need to be created in a nested container-object. This is the object used to create your chronological hypotheses. To create a person-object, you need to hover your mouse over a nested container-object and press the <k-key>.

The person-object will be created in the top-left corner of the current container-object. The object-object contains three text-objects that are color-coded black and red; one control-buttons color-coded gray; and two node-points color-coded yellow and green.



Person-object, "Edit," gray button

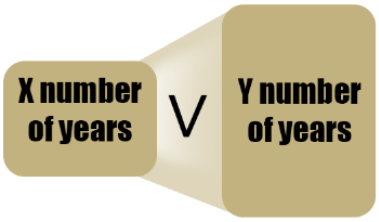
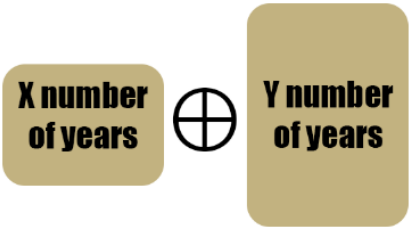
The gray button (top right of the object) is used to edit the text fields of the person-object. When you left-click on the "edit" button, a menu will appear as follows:



There are five fields that can be edited. The *name* field is the identifying information of the person-object. This field does not affect chronological outcomes.

The *reignLength* field is the length the person-object was active. There are three formats available for this field: constant length, a year-range, and a disjunctive range. For the constant length, this is a single integer; e.g., "32" expressing that this king reigned 32 years. For the year-range length, this is a range of reigns; e.g., "20t23" expressing that this king reigned from 20 to 23 years. For the disjunctive length, this is one of two possible reign-lengths; e.g., "3v13" expressing that this king reigned either 3 years or 13 years.

For person-objects, there are only three datatypes: static, inclusive or-conditional, and exclusive or-conditional.

Datatype	Description	Ambiguity[2]	Example	Image
Static	Defines the exact number of years a person reigned	No	**34** "34 years"	
Inclusive or-conditional[3]	Defines a range of years from x to y.	Yes	**10t14** "10 to 14 years"	
Exclusive or-conditional	Defines a reign as either x or y years.	Yes	**10v14** "10 or 14 years"	

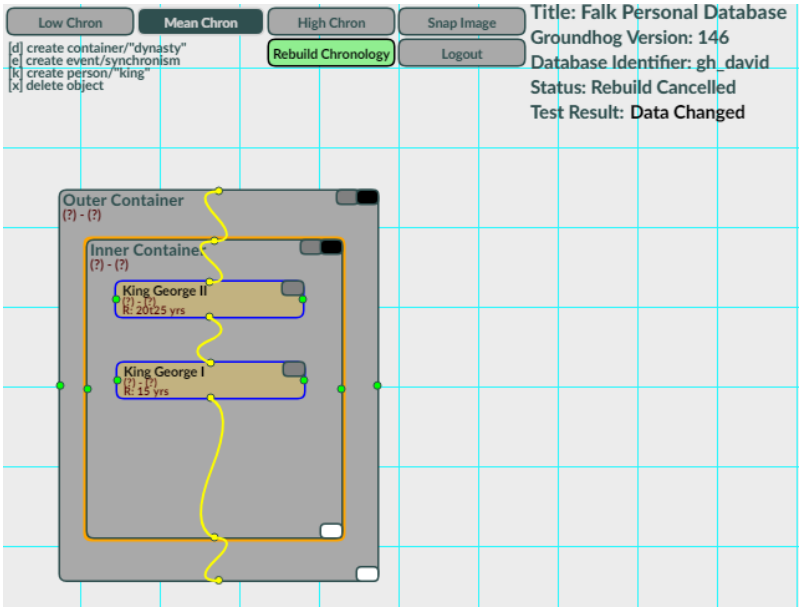
[2] In other words, does the use of this datatype affect (or double) the number of permutations that are being calculated.

[3] It should be noted that, because of abstraction, there is little practical difference between inclusive and exclusive or-conditions until the validation phase of the software.

The *yearsAsCoregent* and *yearType* fields are only used for informational purposes. The *citation* field is used as a note or to cite a source of information.

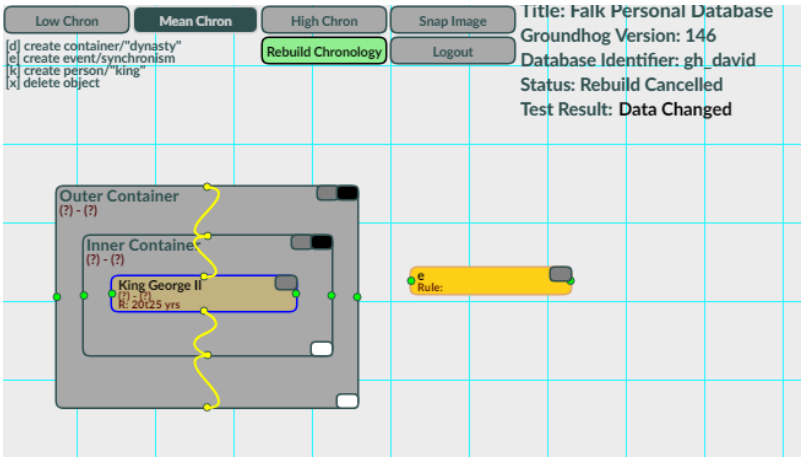
Activating a Person-object

In order to make a person-object active, it has to connected to other objects. Both yellow node-points of a person-object either need to be connected to a container-object or another person object. Connect the top yellow node-point of the outer container to the top yellow node-point of the inner container. To do that left-click one of the node-points and drag it to the other node-point.



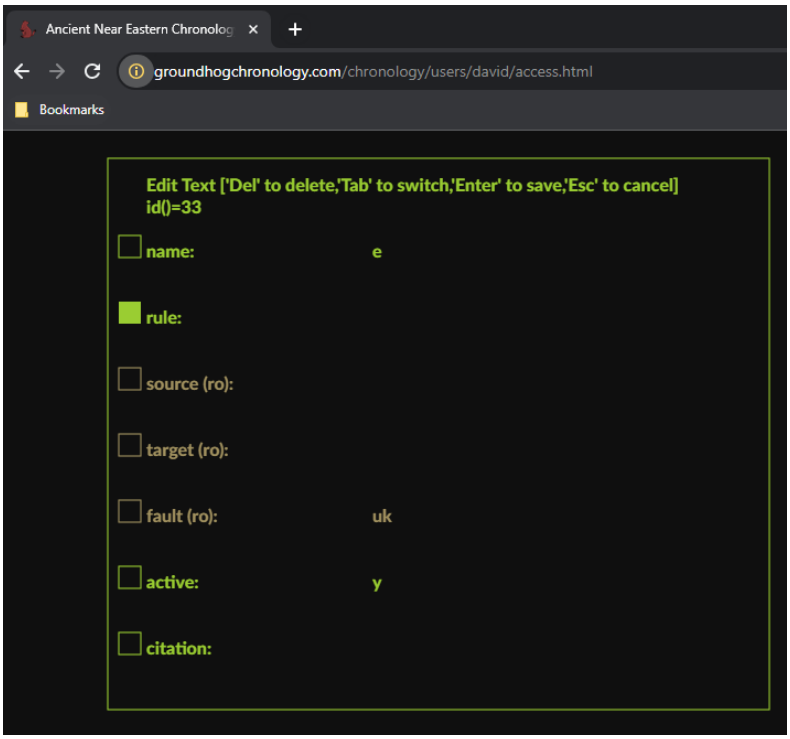
Event-Objects

Event-objects need to be created directly on the workspace. This is the object used to create the synchronisms that are used to test or anchor your hypothesis. To create an event-object, you need to hover your mouse over an empty part of the workspace and press the <e-key>.



Event-object, "Edit," gray button

The gray button (top right of the object) is used to edit the text fields of the event-object. When you left-click on the "edit" button, a menu will appear as follows:



The edit screen shows seven fields. The fields in orange [*source (ro), target (ro),* and *fault (ro)*] are read-only fields that cannot be edited but are there for information or diagnostic purposes. The fields in green [*name, rule, active,* and *citation*] can be edited.

The *name* field is the identifying information of the even-object. This field does not affect chronological outcomes. The *citation* field is used as a note or to cite a source of information. The *active* field only takes a "y" or "n" as valid input to activate or deactivate an event. If the event is deactivated, it will not be used when the chronology is rebuilt.

The *rule* field describes the type of synchronism that is being used for this event. This is perhaps the most complex part of the software. While the syntax of the rule is limited to only four fields, this allows for a great deal of flexibility in the expression of data types. We will explain this using multiple methods to make it clear how these rules are applied to synchronisms.

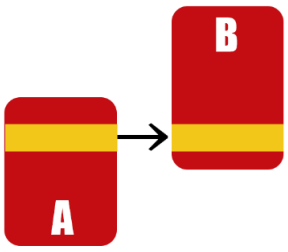
Rule Syntax and Interpretation

Rules used in the *rule* field uses a four-field colon-separated notation. These rules cover a wide range of possible of synchronistic relationships. The notation uses the following fields: **[class]:[king 1]:[king 2]:[year]** . Not all classes of synchronisms use all the fields. The king fields can take on of three formats: the identifier number of the person (e.g., "31" where 31 equals "King Robert"),[4] the identifier anchored to accession year (e.g., "31b0" where this represents the first regnal year of King Robert), and the identifier anchored back from the death year (e.g. "31e1" where this represents the year before King Robert died). For "b" and "e" operators, it should be noted that the count begins at zero.

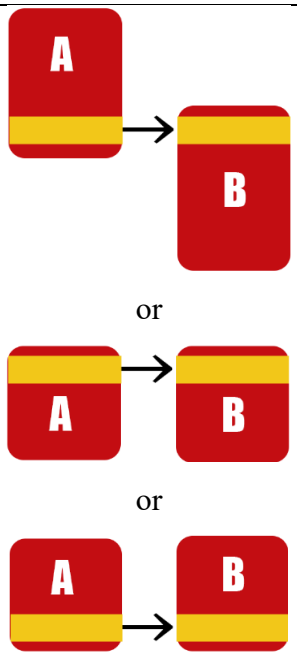
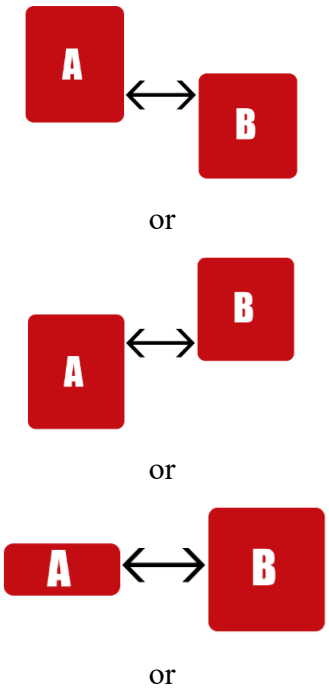
[4] To find out the id number for person-object, hover the mouse reticle over the person-object.

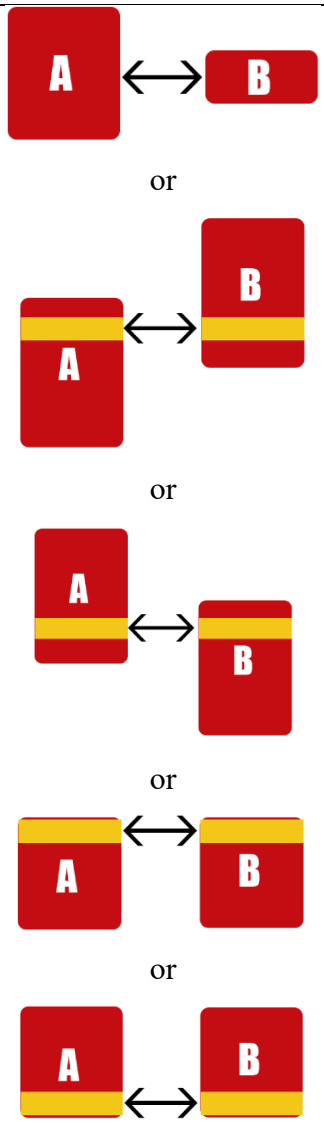
Important Note for Years and Anno Domini Dating

For class 0, 1, 4, 50, and 51 type events, the final field is an absolute calendar year. By default, that year uses the BC system of dating and is expressed as a positive number. Positive numbers were selected for this purpose since most users of *Groundhog* investigate BC dating problems. To use Anno Domini (AD) dating, year numbers must be expressed a negative numbers. The scale for AD dating begins at 0. Therefore, 0 = AD 1, -1 = AD 2, -2 = AD 3, -634 = AD 635, etc. The javascript interface will then correct the resulting dates to canonical dates.

Class	Description	Example	Image	Notations[5]
0	Anchor one regnal year to an absolute calendar year. Third field not used.	**0:31b5::1000** "The sixth year (5+1) of King A (id:31) was in 1000 BCE"		Year = A(y)
1	An absolute year occurs during the reign of King A. "b" and "e" notation is not supported in second field. Third field not used.	**1:31::1000** "1000 BCE occurs during the reign of King A (id:31)"		Year ⊆ A
2	The specific year of King A is equal to the specific year of King B. This is a unidirectional synchronism	**2:21b18:97b1:** "The 19th year of King A (id:21) is equal to the 2nd year of King B (id:97)."	 or	A(y) →= B(y), A →⊤ B, A →⊥ B

[5] Much of this notation has been used from Levy et.al, Table 2, (p. 5), which is based upon Allen's interval algebra (James F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM* 26 [1983] 832-843). Additional notation has been added where Levy et al does not cover the datatypes. The *Groundhog* project has developed its own notation; however, the reader may be more familiar with Allen's or the Levay et al. notation. It should be noted that *Groundhog* event-object rules cover a wider number of situations than are expressed by the Levay et al. notation, which is why multiple Levay et al. notations can be assigned to a single *Groundhog* rule.

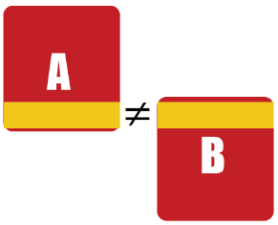
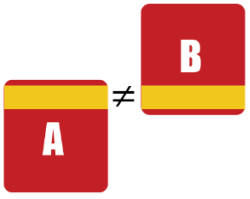
	that can be used for flow control. The second field is the source, and the third field is the target. Hence, A(y) →= B(y), but not A(y) =← B(y). Fourth field not used.			
3	Links two kings as being living during the same time. A class 3 synchronism with both kings defined to exact regnal years is a bidirectional alternative for a class 2 synchronism. Fourth field is not used.	**3:21:91:** "King A (id:21) lived during at least one year of King B (id:91)." **3:21b9:91:** "King B (id:91) lived during the 10th year (9+1) of King A (id:21)."		A ≤ B, A ≥ B, A ← B, A → B, A ⊆ B, A ⊇ B A(y) = B(y), A ⊤ B, A ⊥ B

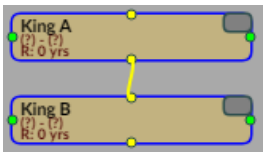
			 A B or B A or A B or A B or A B	

4	An absolute year occurs during the reign of King A. This rule is only used for validation testing and is not used to establish dates. Third field not used.	**4:31::1000** "1000 BCE occurs during the reign of King A (id:31)"		Year ⊆ A
50[6]	Asynchronism[7] to an absolute year.	**50:31b5::1000** "The sixth year (5+1) of King A (id:31) was NOT in 1000 BCE"		Year ≠ A(y)
51	Asynchronism where the absolute year does not occur during the reign of King A.	**51:31::1000** "1000 BCE does not occur during the reign of King A (id:31)"		Year ≠ A

[6] Asynchronisms are not supported by *Chronolog* (Levy et al. 2021,4), as such we have had to augment their notation with our own.

[7] Asynchronisms are not-conditionals or negated synchronisms. So instead of an event occurring in a specific year, with an asynchronism, an event does not occur in that year.

52	Asynchronism where the specific year of King A is not equal to the specific year of King B.	**52:21b18:97b1:** "The 19th year of King A (id:21) is not equal to the 2nd year of King B (id:97)."	 or  or  or 	A(y) ≠ B(y)
53	Asynchronism where the two kings did not live during the same time.	**3:21:91:** "King A (id:21) did not live during the time of King B (id:91)."		A ≠ B
*	A link between two kings where a king precedes a second king.	There is no formal event rule for this. However, the same thing is done manually by connecting one yellow-node to	 or	A(end)=B(start), B(end)=A(start)

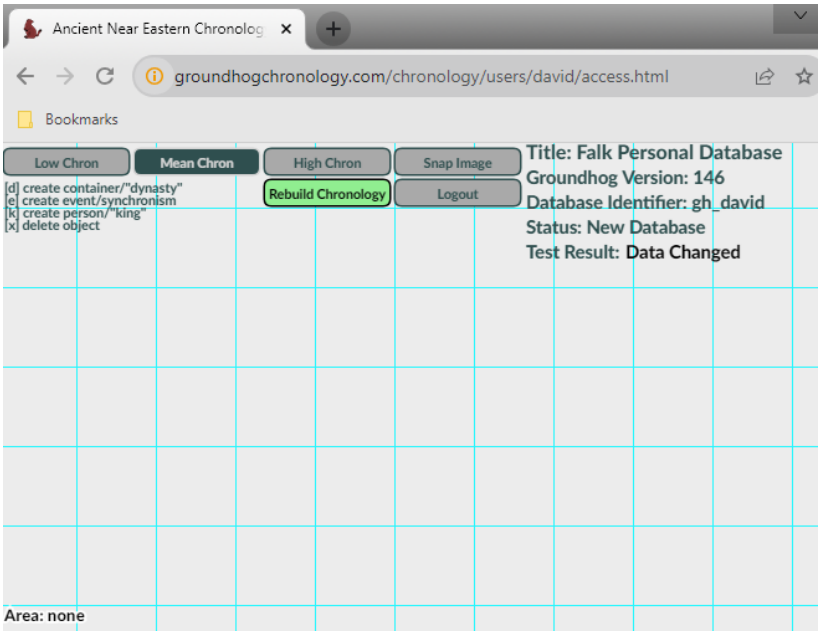
		another yellow-node with a link. For example, 		

Creating Your First Chronology: Step by Step

To create a minimum chronology three elements are required: a nested container-object (a container-object nested within another container-object), a person-object, and an event-object that references an absolute calendar year. However, while that is all is required to create a basic chronology, to create a robust chronology there are recommended practices that will make the data easier to handle and building chronologies more efficient.

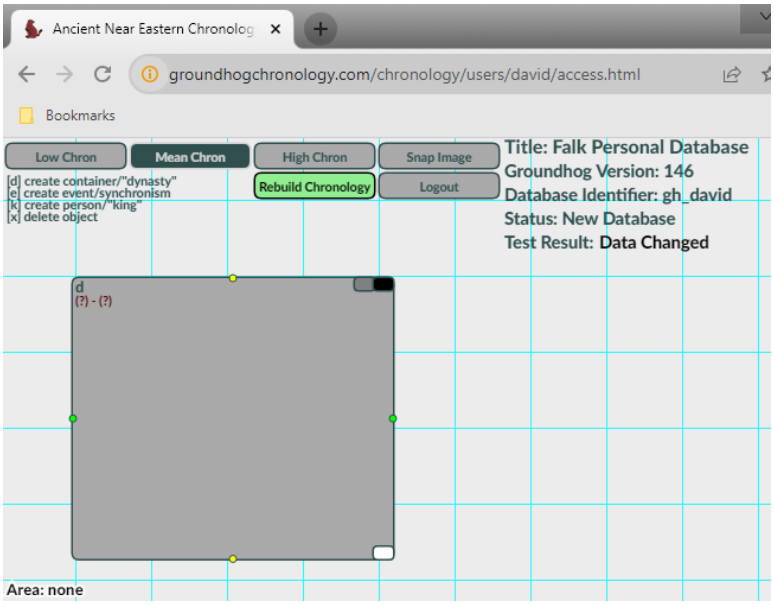
1. Login to your workspace

You need to create container-objects to manage your person-objects.



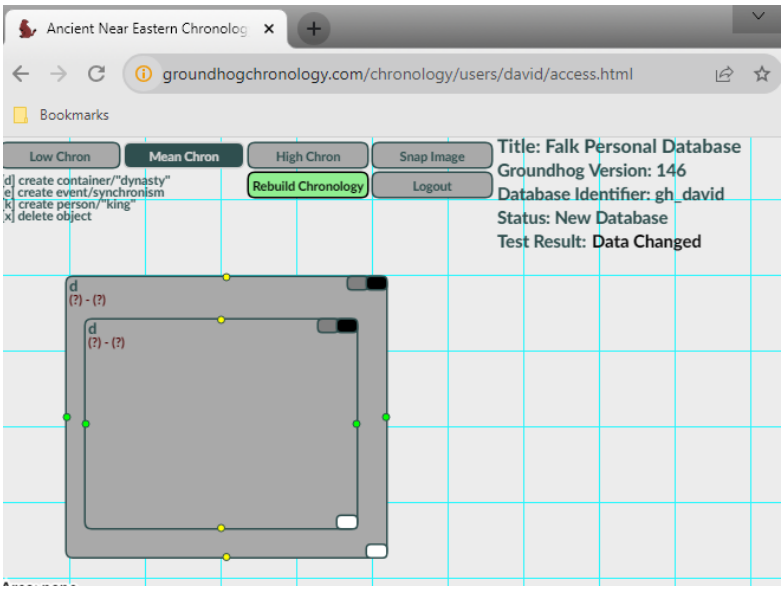
2. Add an initial container-object

You need to create a container-object on your work space.



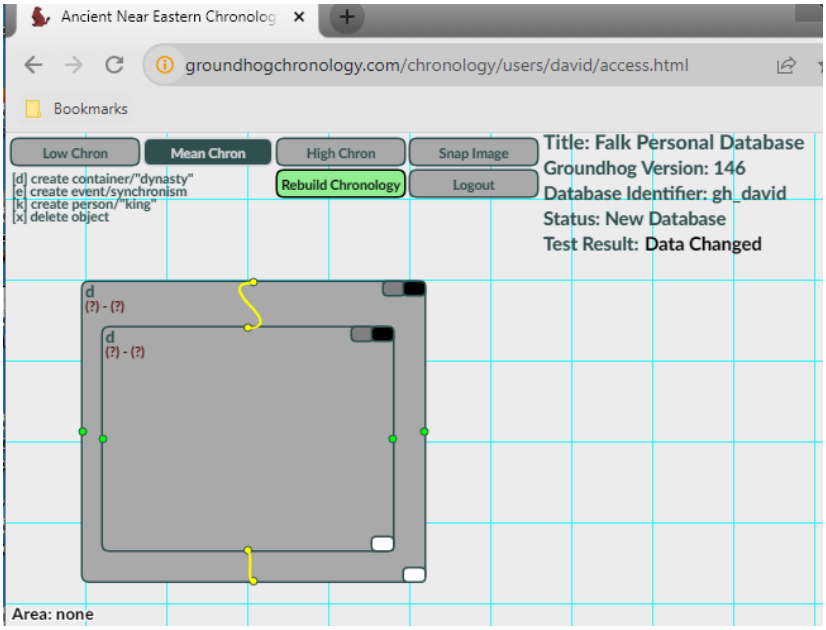
3. Add a second container/"dynasty" object inside the first container

You need to create container-objects to manage your person-objects.



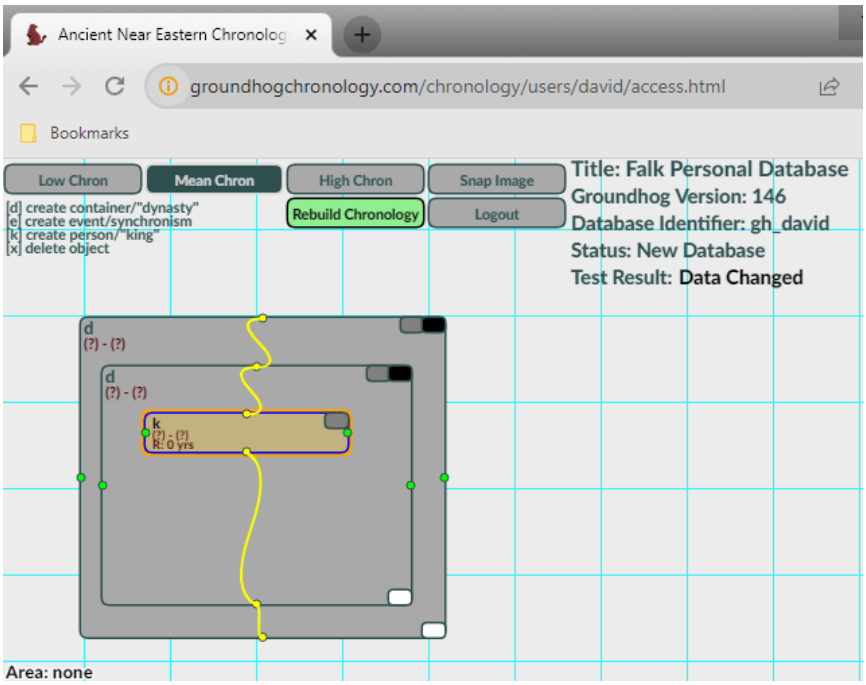
4. Link your container-objects to each other.

Connect the header and footer links together.



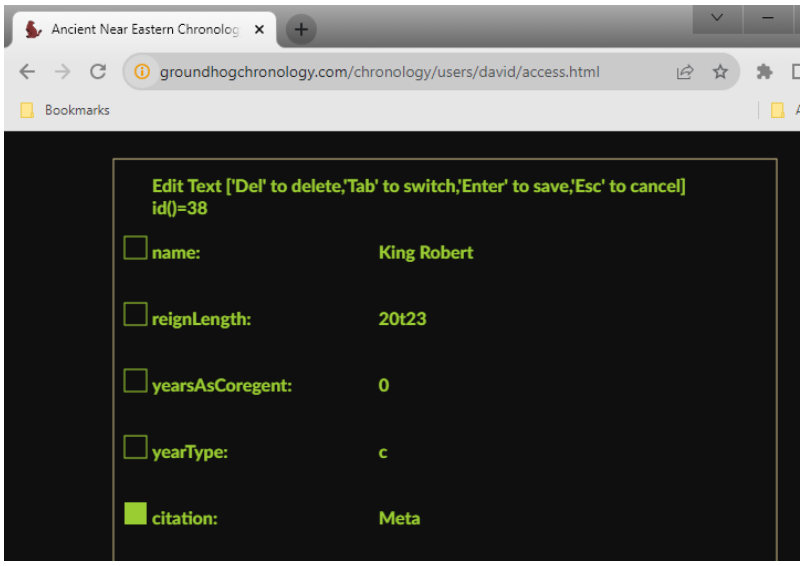
5. Add your first person to the innermost container-object.

Add a person-object to the innermost container and link the person-object to the container-object.



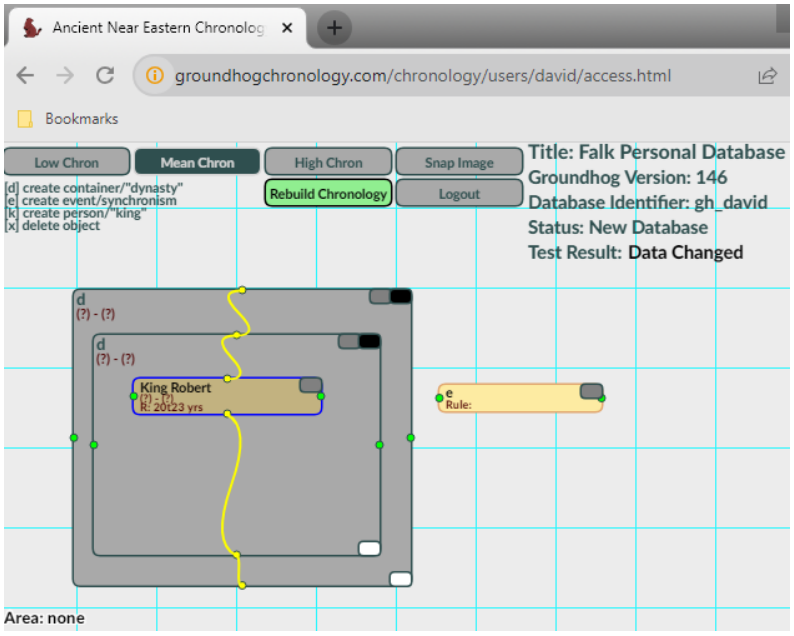
6. Update the information inside the person-object.

Add a person-object to the innermost container and link the person-object to the container-object. When you have finished updating the info, type <enter> to save.



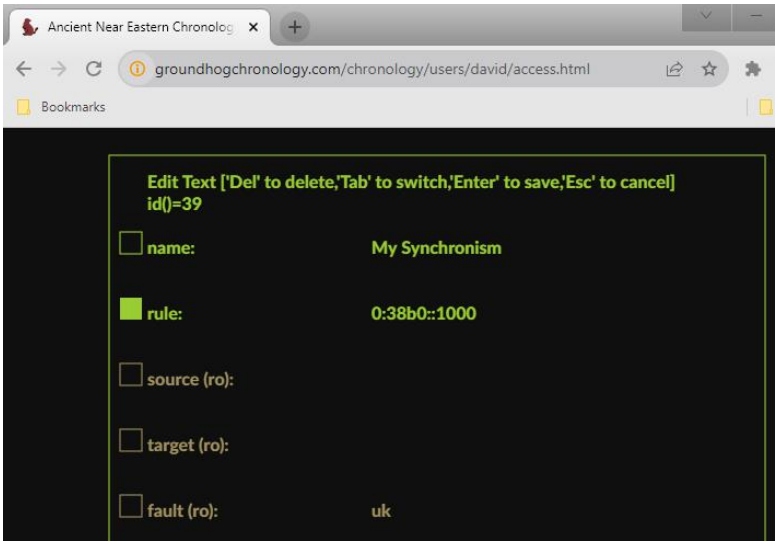
7. Add an event-object.

Add a person-object to the innermost container and link the person-object to the container-object. When you have finished updating the info, type <enter> to save.

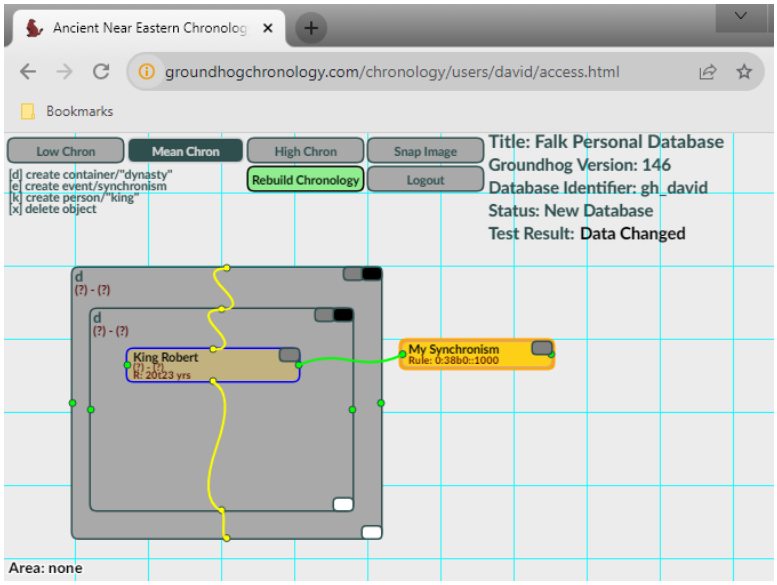


8. Update the event-object fields.

For the event-object, update the *name* and *rule* fields. By hovering over the person-object or by looking at the person-object information display, we can figure out the id number of the person-object (in this case 38). To assign the first year of "King Robert" to the year 1000 BC, we need to add a Class 0 rule to the event, i.e., **0:38b0::1000**. Please note that all chronologies require at least one Class 0 rule to properly build.

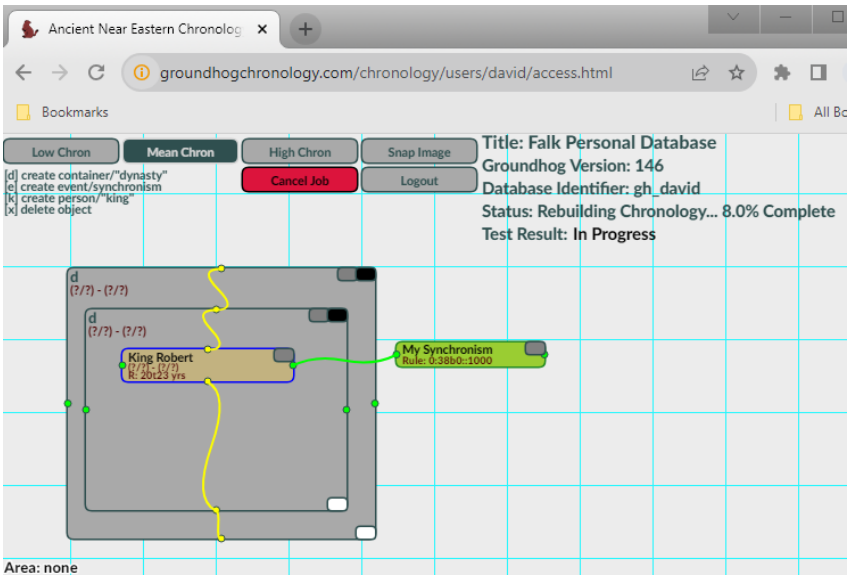


If the rule was entered in with all the correct information, a green link will be attached from the event-object to the correct person-object.



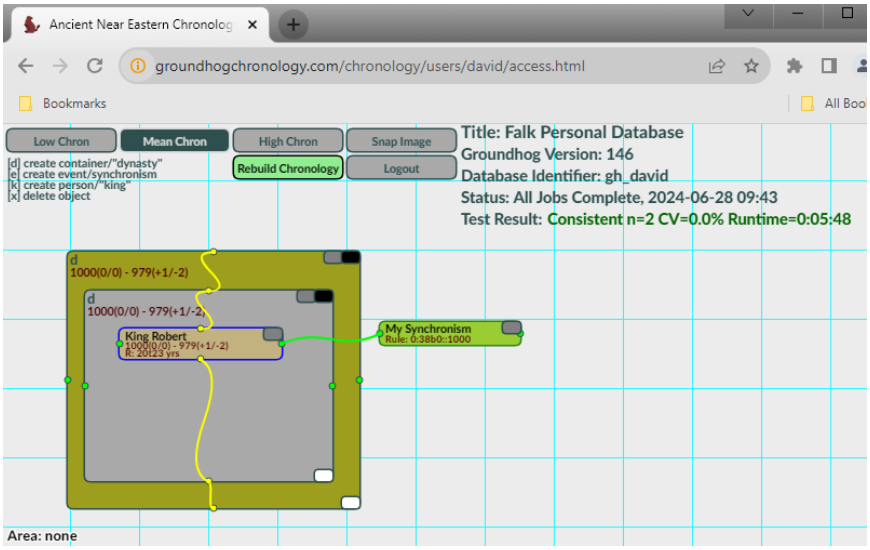
9. Rebuild Chronology

With all the minimal information supplied, you can now click on the <rebuild chronology> button to start rebuilding the chronology. While the chronology is rebuilding, the rebuild bottom will change to a red <cancel job> button. The page will automatically reload and present updates with the progress of the build.



10. Chronology Rebuild Complete

When the rebuild completes, the system will return “Test Result” information as to whether your chronology was consistent or internally inconsistent.

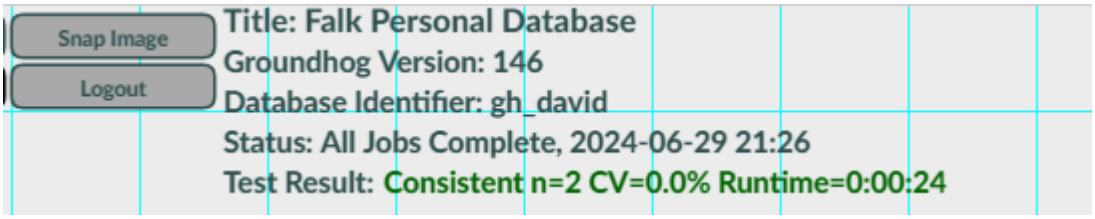


Interpreting the Results

Once you have completed a chronology build and the result is calculated, the workspace will be color-coded. Each of the objects will be color-coded to show where there are inconsistencies.

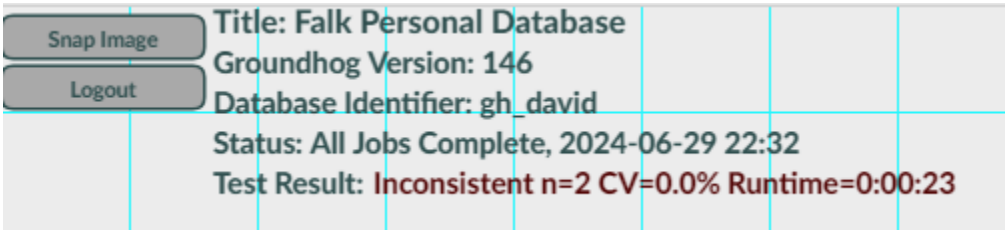
Header Information

In the header, the summary of the overall validation test result will be shown.



In the above example, the "Test Result" provides four pieces of information. If the result is color-coded green, you will also see "Consistent" as the overall test result. You will also see three variables shown. "n" is the number of permutations that were tested prior to any abstraction. "CV" is the coefficient of variation. While the Groundhog does not use a statistical model, such that terms such as standard deviation have no meaning for the result (i.e., the result is internally consistent or it is not), the coefficient of variation has meaning in this model. "Runtime" it the time it took the server to rebuild the chronology.

If at least one permutation is shown to be consistent, the chronology is assigned a value that is internally "Consistent." If all the permutations fail to be internally consistent, then the value is assigned a value that is internally "Inconsistent."



The above is an example of an inconsistent result. The text is marked in red with the word "Inconsistent." In this example, none of the permutations (n) were internally consistent. So, there is no possibility that this chronology could produce a valid result.

If the In the event where the inconsistency is only off by factors of less than two years, which could be caused by rounding errors or the difference between different calendar types, the operator is notified through either an "Inconsistent (marginal=1)" or "Inconsistent (marginal=2)" result. This tells the operator that the chronology is internally inconsistent but only marginally by a factor of 1 or 2 years. This allows the operator to fix the chronology where appropriate and run the test again.

Object Color Information

After rebuilding the objects are color-coded with consistency information.

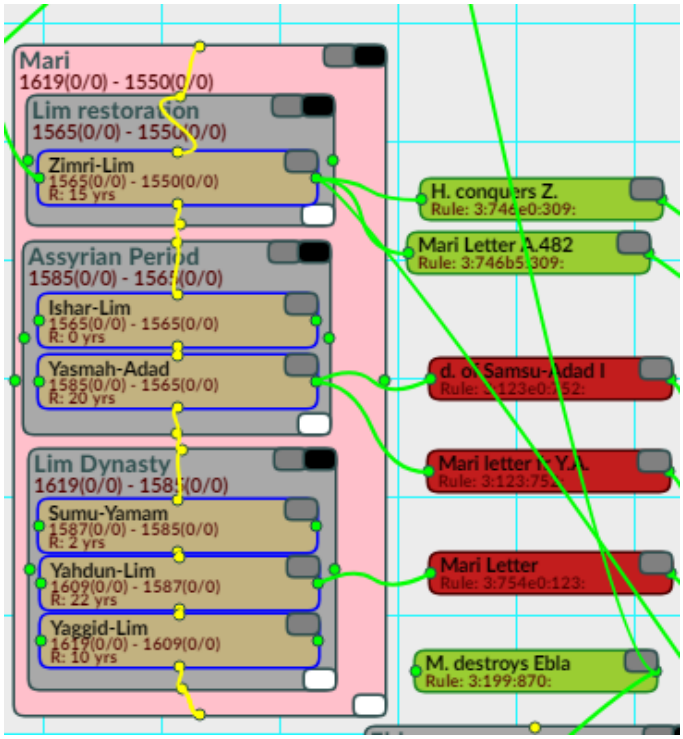
Color Information for Person-Objects

Person-objects are always a beige color, and this color does not change.

Color Information for Container-Objects

With container-objects, the innermost container-object will always be colored grey. If the chronology cannot be derived or the results for that container-objects cannot being computed, the container will also remain grey.

- If at least one of the tested chronological permutations is consistent with the synchronisms, the outer container-object is colored olive green.
- If an error condition results or all the parent chronologies were marked inconsistent, the container-object is marked pink.



Color Information for Event-Objects with Synchronism Rules

Event-objects with synchronism rules have rules that have a class number between 0 and 3.

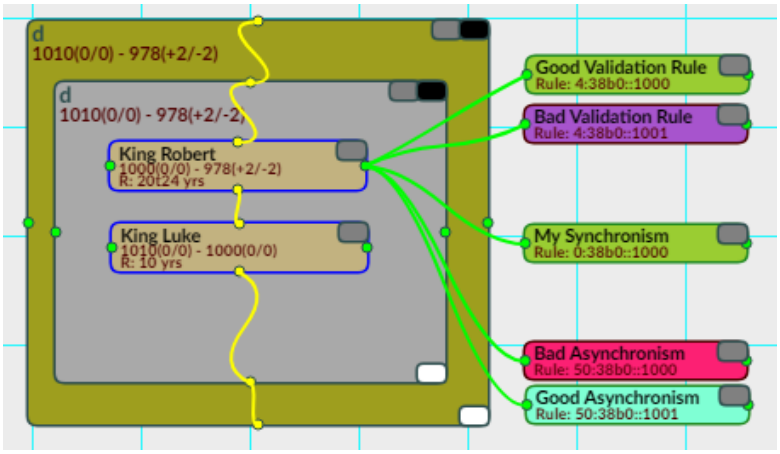
- When a new event-object with synchronism rule is created, and there is no consistency information available, the object is colored yellow. If in the rule specified in the event-object is consistent with the person-object, the event-object is colored yellow green.
- If the event-object has been disabled, the event-object is colored cadet blue.
- If the event-object is inconsistent by +/-1 year, the event-object is colored light orange.
- If the event-object is inconsistent by +/-2 years, the event-object is colored dark-orange.
- If the event-object is inconsistent by more than +/-2 years, the event-object is colored dark red.

Color Information for Event-Objects with Validation-Only Rules

Event-objects with synchronism rules have validation-only rules that have a class number 4.

- When a new event-object with synchronism rule is created, and there is no consistency information available, the object is colored bright yellow.

- If the event-object has been disabled, the event-object is colored cadet blue.
- If in the rule specified in the event-object is consistent with the person-object, the event-object is colored yellow purple.
- If the event-object is inconsistent, the event-object is colored deep purple.



Color Information for Event-Objects with Asynchronism Rules

Event-objects with synchronism rules have rules that have a class number that is between 50-53.

- When a new event-object with synchronism rule is created, and there is no consistency information available, the object is colored light yellow.
- If the event-object has been disabled, the event-object is colored cadet blue.
- If in the rule specified in the event-object is consistent with the person-object, the event-object is colored light teal.
- If the event-object is inconsistent, the event-object is colored bright fuchsia red.

Support

Reporting bugs

If you are reporting a bug, please send the report to zdfalk@gmail.com. Please, include your userid and the database name. Specify the nature of the bug and include screen shots if possible.

This project is currently being maintained by a single person in his spare time, so any bugs will be addressed and fixed only as time permits. In other words, it could take a while.

Special thanks

We want to thank those who have supported this project. We want to thank those who use the project in their academic work. We also want to thank the University of British Columbia that provided a grant that partially funded the original test equipment. We also want to thank DOSarrest Inc who provided some security internet services. We want to also express our gratitude towards those who use the system and those who have helped us to make this a better service.