

## Using the SeriesQL Search Syntax



### Security Note

What you can see and do in this application depends on the security permissions associated with your 25Live user account. If you can't access something you think you should be able to, contact your 25Live Administrator.

SeriesQL (Series25 Query Language) is the internal search syntax of 25Live. Used from the 25Live Quick Search bar, SeriesQL allows you to ask the system a question in the form of fields, operators, and values then receive a response, such as a list of events, locations, or resources that matches your query.

## The SeriesQL Syntax

The built-in query language in 25Live is made up of a sequence of questions. A SeriesQL query utilizes the following parts.

- **Terms:** A term is the first part of a statement and indicates the type of data you are want to query. For example, "name" is a term that indicates you want to query by name. For a list of all available terms, including descriptions, available operators, and example queries, see the following pages:
  - [Event Query Terms](#)
  - [Location Query Terms](#)
  - [Resource Query Terms](#)
  - [Organization Query Terms](#)
  - [Task Query Terms](#)
- **Operators:** An operator follows the term and indicates how you want to compare the term to your results. A certain term only supports certain operators. Examples of operators are =, contains, startsWith, endsWith, between, in, notIn, all. After you enter a term, a suggestion drop-down displays the supported operators for that term for you to select from.
  - The **in**, **notIn**, **all** operators are list operators that can accept a list of values, with each value contained within single-quotes and the whole list surrounded by parentheses. For example:  

```
::state in ('Tentative', 'Confirmed')
```

  
...will return events with a state matching any of those in the value list, in this case, Tentative or Confirmed.
  - The **notIn** operator is the opposite of the in operator. It will return results that do not match any of the values in the values list.
    - *Note: The "notIn" operator is not a valid option for folders when performing event queries.*
  - The **all** operator will return results that match all the values in the list. For example, events can have many categories, but you might want to get events where each event has certain categories:  

```
::category all ('Recital', 'Over 100 Guests', 'Wind Instruments')
```

  
The above query will return events that are in all three categories: "Recital", "Over 100 Guests", and "Wind Instruments".

- **Values:** A value follows an operator and is either a string (enclosed in single-quotes), a date (yyyy-mm-dd format in single-quotes), or a number. The value works with the term and operator to limit results. For example, 'Tentative', 'Confirmed', 'Recital', 'Over 100 Guests', and 'Wind Instruments' are string values. An integer value is a number like 1, 2, 3, etc. A date value needs special formatting as described below.

For even more information, please see the Additional Detail section below.



### You Don't Have to Memorize the Syntax

25Live has built-in help for entering SeriesQL queries. Simply start typing and suggestions will be displayed. After typing or choosing a field item, operator suggestions will display. After typing a value, conjunction suggestions to connect your queries will display.

## To Create a SeriesQL Query

First, navigate to the Search section by using the Go to Search button in the [top navigation bar](#) on every page of 25Live. There is also a Search link in the More menu in the top navigation. The Search section defaults to the Quick Search mode.



### Tip: Please Review Examples

This section provides general information on the steps required to create a query using SeriesQL and examples. Before you begin creating your own queries, you will benefit from reading through all of the following details and examples to learn about all query types and how to best use them to get the information you want. See examples below.

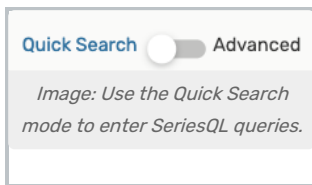
## 1. Choose the Search Type

Using the dropdown menu, you can choose to search for Events, Locations, Organizations, Resources, or Tasks.

The screenshot shows the search interface with the 'Select Object' dropdown menu open. The menu lists five options: Events (with a calendar icon), Locations (with a location pin icon), Organizations (with a group of people icon), Resources (with a diamond icon), and Tasks (with a checklist icon). The 'Events' option is currently selected. To the left of the dropdown, there is a 'Quick Search' toggle switch that is turned on, and a text input field with the placeholder 'Enter Event Search'. Below the input field is a 'More Options' dropdown button. To the right of the 'Select Object' dropdown is a 'Saved Searches (optional)' dropdown button. At the bottom right of the search area are three buttons: 'Reset', 'Save As', and 'Search'. A caption at the bottom of the image reads: 'Image: Use the drop-down menu to choose a search area.'

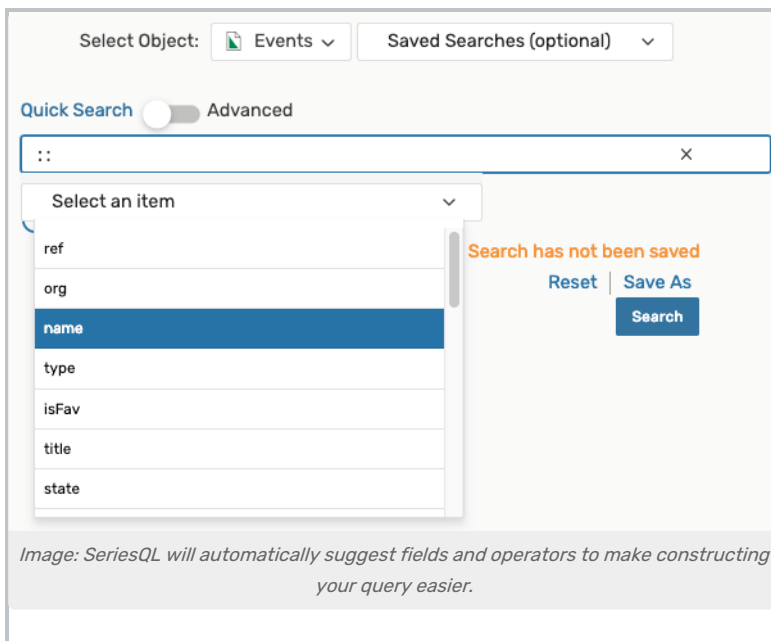
## 2. Be Sure the Search Type is Set to Quick Search

Use the slider control to be sure that you are in Quick Search mode.



### 3. Start Your SeriesQL Query With a Double Colon

Type "::" to start your query statement. You will see a list of fields to choose from.



### 4. Enter or Choose a Term, an Operator, and Value

After entering or choosing a term, SeriesQL will only present you with a list of operators that are appropriate for the term you added. Type or choose the operator you need, then type the value you are looking for. Values must be surrounded by single quotes.



#### Tip: Date Formats

If you are searching for a date value, it must also be enclosed in single quotation marks, but it must also follow a specific format, separated by dashes.

I want to find an event with occurrence date(s) between 11/01/2024 and 12/21/2024.

Example: `::occDate between '2024-11-01' and '2024-12-21'`

If you are adding a time to your date, then you must include it within the single quotations, use a T separator right after the date, and use 24-hour format.

I want to find an event with occurrence date(s) between 11/01/2024 at 6:00 am and 12/21/2024 at 5:00 pm.

Example: `::occDate between '2024-11-01T06:00:00' and '2024-12-21T17:00:00'`

Read additional detail about date values below.

## 5. Optionally Enter or Choose a Conjunction to Add Additional Criteria

You have the option to add additional criteria by entering or choosing a conjunction (i.e. and, or), or you may run the search.

Quick Search ☐ Advanced

::name startsWith 'f' and state in ('Confirmed')

×

?

Hint! Type :: to use SeriesQL.

Search has not been saved

Reset | Save As | Search

List | Calendar

← Future Only →

Recent & Future | Future | All Dates

Select Columns

↻

	Name	Type	Locations	Resources
☆	<a href="#">Foreign Film Festival</a>	Film / Movie	<a href="#">MSC BRB</a>	<a href="#">AV - Blu-ray Player</a> , <a href="#">AV - Data Projector</a> , <a href="#">AV - DVD Player</a> , <a href="#">AV - Extension Cord</a> , <a href="#">AV - Microphone - Hand Held Wireless</a> , <a href="#">AV - Power Strip</a> , <a href="#">AV - Screen - 10'</a> , <a href="#">AV - Sound System</a> , <a href="#">AV - Technician</a> , <a href="#">CS - Event Support Staff</a>
☆	<a href="#">Future Leaders Discussion Group</a>	Meeting	<a href="#">MSC 108</a>	<a href="#">AV - DVD Player</a> , <a href="#">FAC - Board - Marker</a> , <a href="#">SC - Chairs - Stacking</a> , <a href="#">SC - Data Projector</a> , <a href="#">SC - Screen - 10'</a>

Image: This query is composed of two statements joined by "and." It finds all events that begin with "f" and are in a "Confirmed" event state.

6. Run the Search

Use the **Search** button or your Enter or Return key to view [your search results](#).

Additional Detail: Anatomy of a SeriesQL Query

A query is a series of statements joined together by 'and' and 'or' operators. A statement consists of a **term**, an **operator**, and a **value or list of values** for certain operators (in, notIn, all). When executed, the query returns a list of matching results. Each item returned in the results is true with respect to the query. Let's take a look at a SeriesQL example for events.

```
::name contains 'audition'
```

This query asks for all events containing the word "audition". The double colons ("::") tell the system that you are making a SeriesQL query and not just a keyword search. The word "name" is a term, and since this is an event search, it means you are asking a question about event names. The word "contains" is an operator and means you want to search for events with names that contain some value. The value 'audition' is within single-quotes, indicating it is a value and that you want events with names containing the word "audition".

You can compose queries by adding additional statements. Here is an example of a multi-statement query:

```
::(name contains 'checkers' or name contains 'chess') and missingLocationAssn between today and start + 1
```

This query asks for all events that contain either the word "checkers" or the word "chess" and also have a missing location assignment between today and tomorrow. There are three statements in this query:

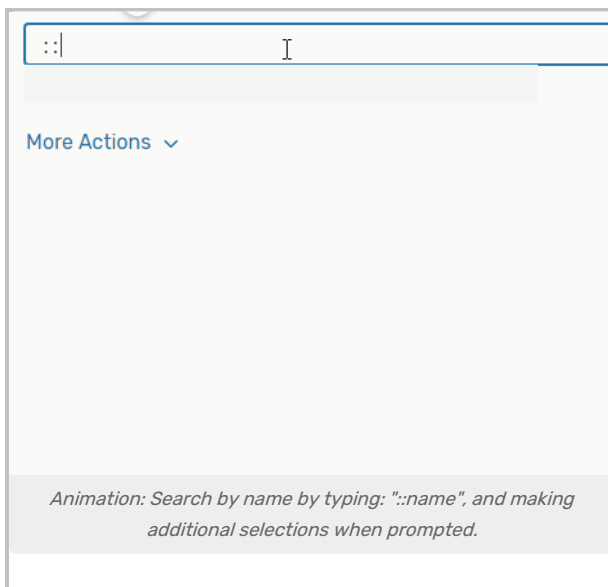
- 1. name contains 'checkers'
- 2. name contains 'chess'
- 3. missingLocationAssn between today and start + 1

The first two statements are grouped together with parentheses to protect against the 'and' operator over-selecting results (see below). This grouping ensures that the last statement is applied after the first two statements have been checked because we want the missing location assignments from today and tomorrow to apply to both checkers events and chess events. The "or" operator joins the checkers and chess statements so we'll get events with names that have the words 'chess' or 'checkers' in them. The last statement limits the checkers/chess events we get to just those that have missing location assignments between today and tomorrow. Note that for date terms, you can use the special "today" value to refer to today's date. SeriesQL also supports date math, so you can add or subtract from "today" in your queries.

### SeriesQL Examples

#### Example 1: Simple Single Criteria

Here the user is creating a simple one-statement query to find all events named "foot".



The screenshot shows a query editor interface. At the top, there is a text input field containing the query fragment `::|` followed by a cursor. Below the input field is a button labeled "More Actions" with a downward-pointing chevron. At the bottom of the editor, there is a light gray footer area containing the text: *Animation: Search by name by typing: "::name", and making additional selections when prompted.*

#### Example 2: Choosing Multiple Items in Single Criteria

Here the user is creating an event query for events matching all specified categories. Note that after the user chooses "category" and "all" they can select the categories they want and then click Done to insert them into the query.

The screenshot shows a search interface with a text input field containing the text "::category". Below the input field is a dropdown menu titled "Select an item" with a downward arrow. The dropdown menu is open, showing a list of options: "cabinet", "contact", "category", "custAtrb", "ref", "org", and "name". A blue letter "M" is visible to the left of the dropdown. Below the dropdown menu, there is a grey box with the text: "Animation: Search by category by typing "::category", and making additional selections when prompted."

### Example 3: Multiple Option Choices in Criteria

Here the user is creating an event query for events that are in at least one of the listed locations. Note how the user can search locations dynamically by typing, using arrows to navigate to locations, entering to select them, and finally clicking Done to insert them into the query.

The screenshot shows a search interface with a search bar containing the text "Enter Locations Search". Above the search bar are two tabs: "Quick Search" (selected) and "Advanced". Below the search bar is a hint icon and the text "Hint! Type :: to use SeriesQL". Below the hint is a dropdown menu labeled "More Options" with a downward arrow. Below the dropdown menu, there is a grey box with the text: "Animation: Search by location by typing "::location", and making additional selections when prompted."

### Example 4: Searching By Location Preference

A user can also search by space (location) preference using

The screenshot shows a search interface with a search bar containing the text "::locationPreferences in ('BCC 413 --- Baker Classroom Complex - Room 413 -- Cap: 35', 'BCC 415 --- Baker". Below the search bar, there is a grey box with the text: "Image: Users can search by location preference."

To begin, use the syntax, `::locationPreferences in ('search term')`. Then add details about a location.

## Additional Detail: More About Values

## Date Values

Date values in SeriesQL need to be entered in yyyy-mm-dd format. This means 4 digit year, two-digit month (01, 02, ..., 10, 11, 12), and two-digit day (01, 02, ..., 10, etc). To specify date and time, simply enter a date in the correct format, add an uppercase T, and enter the time in hh:mm format, which means two-digit hour in 24-hour format (00, 01, ..., 11, 12, 13, 14, 15, ..., 23), and two-digit minute (00, 01, ..., 59). For midnight, use 23:59.

You may also use the special value, **today**, to indicate today's date when using a range.

Dates also support arithmetic operations so you can add and subtract from them. For example, today - 1 is yesterday and today + 1 is tomorrow. Similarly, '2024-05-04' - 4 is '2024-04-30'.

### Date Examples:

```
::earliestStart = '2024-05-04'
```

```
::earliestStart = '2024-05-04' - 4
```

### Date/Time Examples:

```
::occDate between '2024-04-19T00:00' and '2024-04-19T14:00'
```

```
::occDate between '2024-04-19T00:00' and today
```

## Embedded Values

An embedded value is a value the system inserts into the beginning of an 'in' clause because it needs it. There can only be one embedded value in an 'in' clause and it is always at the beginning. The actual value can be ignored because the system sets it for its own needs to execute or switch to design view properly. Different terms can have different embedded values. Only two terms have embedded values:

1. contact → embedded value is a contact role
2. relationship → embedded value is a relationship type

Examples include:

```
::contact in ('role:-1', 'Alice')
```

In the event query example above, 'role:-1' is an embedded value that indicates 'Requestor'. You can see this by switching to design view and seeing that Requestor is selected. If you know the embedded value you want, you can edit the SeriesQL directly. For example, you could change -1 to -3, which would change the contact type in the query from Requestor to Instructor.

```
::relationship in ('relationship:4', 'Conference Room A', 'Conference Room B')
```

In the location query example above, the query is asking for locations that are related, by relationship type id 4, to the selected locations: 'Conference Room A' and 'Conference Room B'. In this case, relationship type id 4 corresponds to 'Also Assign'. You can easily see what each embedded value corresponds to by switching to the design view.

When selecting actual values (contacts, locations, or resources) for a term with an embedded value, search for and select your contacts or locations or resources first, as if the embedded value doesn't exist. After you select values in the suggestion drop-down, select the single contact role or relationship type you want and click Done. The system will fill the 'in' clause for you with the selected embedded value as well as your selected actual values. Below is a visual example – note that the user just searches in the text area, selects what they want, selects the embedded value they want, and clicks Done.



Enter Search

*Hint! Type :: to enter Advanced Query mode. Use Enter to run the search.*

More Actions

lity

*Animation: Search by relationship by typing "::relationship", and making additional selections when prompted.*

## Custom Attribute Value

Custom attributes are special in that there are many different types of them and, as a result, they can take on different values depending on the type. When you search by custom attribute, you are searching for results that have the selected custom attribute and match any value you have specified. In SeriesQL, this is represented as a list of values inside an "in" operator. For most custom attributes, you have to specify a comparison operator, also known as a relationship in this context, like "starts with" or "is earlier than or equal to" and a value, like "computational theory" or "2024-01-01". Note that when you enter a value via the suggestion drop-down, you don't need to include quotes because the system does that for you after you click the "Done" button. Most custom attributes need values to form a search, but a few don't. For example, boolean (true/false) custom attributes don't require a value. Similarly, if your comparison operator is "exists" or "does not exist", you are asking for results where the selected custom attribute exists (or does not exist), so no value is needed.

Examples of custom attribute event queries:

```

::custAtrb in ('Cancel if Bad Weather?', 'is True', 'B')
::custAtrb in ('Course Section', 'starts with', 'S', 'computational theory')
::custAtrb in ('Date Time', 'is earlier than or equal to', 'E', '2020-12-01')
::custAtrb in ('Amount Received', 'exists', 'F')
    
```

Notice in the example queries above that each "in" clause has a single character value – 'B', 'S', 'E', and 'F' in these examples. This value represents the type of custom attribute the system will use when executing the query, as explained below.

- **B: Boolean** --> no value needed, the comparison operator (relationship) defines the value for you, such as 'is True' or 'is False'
- **I: Image** --> no value needed, only exists / does not exist

- X: **Large Text** --> no value needed, only exists / does not exist
- 2: **Organization** --> needs a value if using the 'is equal to' comparison operator. To use, search for and select an organization from the suggestion drop-down
- 3: **Contact** --> needs a value if using the 'is equal to' comparison operator. To use, search for and select a contact from the suggestion drop-down.
- 4: **Location** --> needs a value if using the 'is equal to' comparison operator. To use, search for and select a location from the suggestion drop-down.
- 6: **Resource** --> needs a value if using the 'is equal to' comparison operator. To use, search for and select a resource from the suggestion drop-down.
- R: **File Reference** --> needs a value if using the 'contains', 'is equal to', 'starts with' comparison operators
- S: **String** --> needs a value if using the 'contains', 'is equal to', 'starts with' comparison operators
- D: **Date** --> needs a value if using the 'is equal to', 'is earlier than or equal to', 'is later than or equal to' comparison operators
- E: **Datetime** --> needs a value if using the 'is equal to', 'is earlier than or equal to', 'is later than or equal to' comparison operators
- T: **Time** --> needs a value if using the 'is equal to', 'is earlier than or equal to', 'is later than or equal to' comparison operators
- N: **Integer** --> needs a value if using the 'is equal to', 'is less than or equal to', 'is greater than or equal to' comparison operators
- F: **Float** --> needs a value if using the 'is equal to', 'is less than or equal to', 'is greater than or equal to' comparison operators

### Multiple Suggestion Searches for 'in' Values

Some terms require you to search for values, like the location term in an event search, because there are so many values that the system can't list them all for you in the browser. In other cases, you may want values that are so different that you need to find them separately to include them in your query. For example, say you want all events that are either in a certain conference room or in the gym. To do this, you would search for conference rooms and select the one you want in your query via the suggestion drop-down, and click Done. Next, you would click at the end of the last value inserted (between the last single-quote and end-parentheses), add a comma and a starting single-quote, search for and select the gym via the suggestion drop-down, and click Done to include it in your query.

Enter Search

Hint! Type :: to enter Advanced Query mode. Use Enter to run the search.

More Actions ▾

Animation: Search by space by typing "::space", and making additional selections when prompted.