IQL is a logical language that can be used to return a list of Insight objects that match the statement. IQL can be very simple, or you can use variables, operators, and functions to create a dynamic and powerful query. In this section, you will learn the general syntax of IQL queries and how to use their functions.

**General syntax**
A standard IQL query includes an attribute, an operator, and a value or function:

```
<attribute> <operator> <value/function>
```

Dot-notation can also be used to reference the attributes of objects.

```
<attribute>.<attribute> <operator> <value/function>
```

- Attributes in IQL are case-sensitive.
- Attributes containing spaces must be surrounded by quotes.
- If you are searching for an object that itself contains quotes, you must escape the quotes using a forward slash. For example, if you were searching for a 15-inch (") screen, you would search for:

```
Name = "15\" Screen"
```

**Ordering IQL results**
You can order the results of your IQL query using the following syntax:

```
order by [AttributeName|label] [asc|desc]
```

- If the attribute specified in the order by clause is of the object reference type you can use dot notation to order by attributes on the referenced object. This can be done in unlimited depth. Note that the every dot in the order by clause will decrease the performance of that particular IQL.

```
order by "My object reference"."Attribute on referenced object" asc
```

- Values that are missing will be considered to be last (using asc) or first (using desc).
- The attribute name specified in the IQL must exist in Insight if not the IQL will be considered invalid. The attribute name is case sensitive.
- If the filtered objects does not contain the attribute specified in the order by clause the order of the objects returned will be arbitrary.

**Placeholders**
Placeholders are used to refer to an object type, object, or attribute.

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectType</td>
<td>Limit the search result to a specific object type.</td>
</tr>
</tbody>
</table>

```
objectType = Servers
```
<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td>The value is equal (not case sensitive) to input.</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>The value is less than input.</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>The value is more than input.</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>The value is less or equal than value.</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>The value is more or equal than value.</td>
</tr>
<tr>
<td><code>like</code></td>
<td>The input is a subset of value.</td>
</tr>
<tr>
<td><code>not like</code></td>
<td>The input does not contain the value.</td>
</tr>
<tr>
<td><code>in</code></td>
<td>The input is a set of values.</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>The value is not equal to input.</td>
</tr>
<tr>
<td><code>not in</code></td>
<td>The input is not a set of values.</td>
</tr>
<tr>
<td><code>startswith</code></td>
<td>The input starts with input.</td>
</tr>
<tr>
<td><code>endswith</code></td>
<td>The input ends with input.</td>
</tr>
</tbody>
</table>
is

The input is something or not. It is followed by "empty" or "not empty". It represents if a value is existing or not.

Manager IS NOT EMPTY

"License Date" IS EMPTY

The attributes reference attribute is queried.

==

The value is strict (case sensitive) equal to input.

having

The input is having a property like inbound references.

not having

The input is not having a property like outbound references.

Logic operator

Description

AND

Combines IQL expressions where both parts of the expression must apply.

OR

Combines IQL expressions where either part of the expression may apply.

Date-time functions

Date-time functions can prove very useful in IQL. The following functions are available.

<table>
<thead>
<tr>
<th>Date-time functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>now()</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>startOfDay()</td>
<td>Returns the start time of the current day.</td>
</tr>
<tr>
<td>endOfDay()</td>
<td>Returns the end time of the current day.</td>
</tr>
<tr>
<td>startOfWeek()</td>
<td>Returns the start time of the current week.</td>
</tr>
<tr>
<td>endOfWeek()</td>
<td>Returns the end time of the current week.</td>
</tr>
<tr>
<td>startOfMonth()</td>
<td>Returns the start time of the current month.</td>
</tr>
<tr>
<td>endOfMonth()</td>
<td>Returns the end time of the current month.</td>
</tr>
<tr>
<td>startOfYear()</td>
<td>Returns the start date of the current year.</td>
</tr>
<tr>
<td>endOfYear()</td>
<td>Returns the end date of the current year.</td>
</tr>
</tbody>
</table>

We use m for minutes, h for hours, d for days and w for weeks to represent relative time, e.g. Created > "-2h 15m" returns all objects created in the last 2 hours and 15 minutes.

User functions

User functions return information linked to a user, reporter, or multiple users.

The currentUser() and currentReporter() functions do not currently work as part of an Asset Type Configuration.

<table>
<thead>
<tr>
<th>User Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>currentUser()</td>
<td>Filter on user attributes connected to the current (logged in) user. Attribute needs to be of type User.</td>
</tr>
<tr>
<td>currentReporter()</td>
<td>Filter on user attributes connected to current reporter in custom fields. Attribute needs to be of type User.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>user(user1, user2, ...)</td>
<td>Filter on user attributes having the specified users.</td>
</tr>
<tr>
<td>User in group(&quot;jira-users&quot;, &quot;jira-administrators&quot;)</td>
<td></td>
</tr>
<tr>
<td>User having user(&quot;admin&quot;)</td>
<td></td>
</tr>
</tbody>
</table>

**Group functions**

Group functions return information linked to one or more groups.

<table>
<thead>
<tr>
<th>Group Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group(group1, group2, ...)</td>
<td>Filter on any object connected to a user within a specific group.</td>
</tr>
<tr>
<td>Group having user(&quot;currentReporter()&quot;)</td>
<td></td>
</tr>
<tr>
<td>user(user1, user2, ...)</td>
<td>Filter on any object connected to a user within a specific group.</td>
</tr>
</tbody>
</table>

**Project functions**

Project functions return information linked to the current project. This only works within the context of a ticket.

<table>
<thead>
<tr>
<th>Project Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>currentProject()</td>
<td>Filter on any object connected to the current project in a custom field. Works only in the context of a ticket.</td>
</tr>
<tr>
<td>Project = currentProject()</td>
<td></td>
</tr>
</tbody>
</table>

**Reference functions**

The reference functions can take two arguments: an IQL and/or Reference Types Names. The IQL argument can be an arbitrary IQL including an IQL with reference function. The Reference Type argument is optional.

<table>
<thead>
<tr>
<th>Reference Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inboundReferences(IQL)</td>
<td>Filter objects having inbound references where the objects referencing to the object matches the IQL, e.g. object having inboundReferences() filters all objects having inbound references since the empty IQL will match all objects. If the query is changed to object having inboundReferences(Name=&quot;My Name&quot;) all objects that are referenced by an object with the name &quot;My Name&quot; are filtered.</td>
</tr>
<tr>
<td>outR(IQL)</td>
<td>Filter objects having outbound references where the referenced objects match the IQL, e.g. object having outboundReferences() filters all objects having outbound references since the empty IQL will match all objects.</td>
</tr>
</tbody>
</table>
inboundReferences (IQL, referenceTypes)

Filter object having inbound references where the objects referencing to the object matches the IQL. E.g. object having inboundReferences() will filter all objects having inbound references since the empty IQL will match all objects. If the query is changed to object having inboundReferences(Name="My Name") all objects that is referenced by an object with the name My Name will be filtered.

The Reference Type argument can be simple or advanced.

simple can take only one Reference Type name as parameter and looks like "Installed", e.g. object having inR("", "Installed")

advanced has the following structure: argumentName + operator + ("value 1", "value 2", "value 3"), where
- argumentName: refType or referenceType
- operator: IN or NOT IN

outboundReferences (IQL, referenceTypes)

Filter object having outbound references where the referenced objects match the IQL. E.g. object having outboundReferences() will filter all objects having outbound references since the empty IQL will match all objects.

The Reference Type argument can be simple or advanced.

simple can take only one Reference Type name as parameter and looks like "Installed"

e.g. object having outR("", "Installed")

advanced has the following structure: argumentName + operator + ("value 1", "value 2", "value 3"), where
- argumentName: refType or referenceType
- operator: IN or NOT IN

Connected tickets function

The connectedTickets() function is used to filter objects having tickets connected to them. Specific Jira issues may be selected by providing a proper Jira Query Language (JQL) query. If no JQL query is provided, all objects having Jira issues connected are returned.

<table>
<thead>
<tr>
<th>Connected tickets function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectedTickets()</td>
<td>All objects having tickets connected to them are returned.</td>
</tr>
<tr>
<td>object having connectedTickets()</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>connectedTickets (JQL)</th>
<th>Object having tickets connected to them that match given JQL query are returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>object having connectedTickets(labels is not empty)</td>
<td></td>
</tr>
</tbody>
</table>

Please observe that this powerful function may produce results that are not obvious at the first sight especially when complex queries, that uses for example negation, are constructed. To help interpret them one should think about the query as a two step action: 1. JQL execution 2. IQL execution operating on the JQL result.

Object type functions

These functions return a set of all object and their children based upon the object type name or ID.

<table>
<thead>
<tr>
<th>Object type function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object having connectedTickets()</td>
<td></td>
</tr>
<tr>
<td>objectTypeAndChildren (Name)</td>
<td>Filter objects based on the object type specified by the Name and its children.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><code>objectType in objectTypeAndChildren(&quot;ObjectName&quot;)</code></td>
</tr>
<tr>
<td>objectTypeAndChildren (ID)</td>
<td>Filter objects based on the object type specified by the ID and its children.</td>
</tr>
</tbody>
</table>