IQL - Insight Query Language

The Insight Query Language is something that you can use in different places in Insight. The IQL will also be improved more and more and used in more places within Insight. Below, we will explain how to use it and give you some examples.

Syntax

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

Expression containing space has quote surrounded like "Mathias Edblom"

⚠️ The attribute in the IQL is case sensitive!

Attributes

Placeholders

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectType</td>
<td>You can limit the search result to a specific object type, e.g. objecttype = Servers</td>
</tr>
<tr>
<td>objectTypeId</td>
<td>You can limit the search result on object typ ids like objectTypeId in (1, 2)</td>
</tr>
<tr>
<td>anyAttribute</td>
<td>You can search on any attribute like: &quot;anyAttribute = 123.123.123.123&quot;</td>
</tr>
<tr>
<td>object</td>
<td>Limit the search to the object, e.g. &quot;object having inboundReferences()&quot;</td>
</tr>
<tr>
<td>objectId</td>
<td>Find an object by object schema ID, e.g. &quot;objectId = 114&quot;</td>
</tr>
</tbody>
</table>

Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUAL</td>
<td>=</td>
<td>The value is equal (not case sensitive) to input</td>
</tr>
<tr>
<td>LESS_THEN</td>
<td>&lt;</td>
<td>The value is less then input</td>
</tr>
<tr>
<td>MORE_THEN</td>
<td>&gt;</td>
<td>The value is more then input</td>
</tr>
<tr>
<td>LESS_THEN_AND_EQUAL</td>
<td>&lt;=</td>
<td>The value is less or equal then value</td>
</tr>
<tr>
<td>MORE_THEN_AND_EQUAL</td>
<td>&gt;=</td>
<td>The value is more or equal then value</td>
</tr>
<tr>
<td>LIKE</td>
<td>like</td>
<td>The input is a subset of value</td>
</tr>
<tr>
<td>NOT_LIKE</td>
<td>not like</td>
<td>The input does not contain the value</td>
</tr>
<tr>
<td>IN</td>
<td>in</td>
<td>The input is a set of values</td>
</tr>
<tr>
<td>NOT_EQUAL</td>
<td>!=</td>
<td>The value is not equal to input</td>
</tr>
<tr>
<td>NOT_IN</td>
<td>not in</td>
<td>The input is not a set of values</td>
</tr>
<tr>
<td>STARTS_WITH</td>
<td>startswith</td>
<td>The input starts with input</td>
</tr>
<tr>
<td>ENDS_WITH</td>
<td>endswith</td>
<td>The input ends with input</td>
</tr>
<tr>
<td>IS</td>
<td>is</td>
<td>The input is something or not. It followed by &quot;empty&quot; or &quot;not empty&quot;. It represents if a value is existing or not.</td>
</tr>
<tr>
<td>OUTBOUND_REFERENCE</td>
<td>.</td>
<td>The attributes reference attribute is queried</td>
</tr>
<tr>
<td>STRICT_EQUAL</td>
<td>==</td>
<td>The value is strict (case sensitive) equal to input</td>
</tr>
<tr>
<td>HAVING</td>
<td>having</td>
<td>The input is having a property like inbound references</td>
</tr>
<tr>
<td>NOT HAVING</td>
<td>not having</td>
<td>The input is not having a property like outbound references</td>
</tr>
</tbody>
</table>
Combination
You may use AND/OR to combine IQL expressions.

Functions
Functions are used as the value in the above IQL syntax. They supply dynamic values.

Date:

- now()
- startOfDay()
- endOfDay()
- startOfWeek()
- endOfWeek()
- startOfMonth()
- endOfMonth()
- startOfYear()
- endOfYear()

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.

You may do queries as iqlFunction([offset]), e.g startOfDay(-3w). You may also check for an upcoming date, e.g. now(100d), which could be used for example to see if a license expires within certain amount of time.

IP Address:

- CIDR(IP RANGE) - Filter on IP ranges

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;IP Address&quot; IN CIDR(&quot;192.0.0.0/8&quot;)</td>
</tr>
<tr>
<td>&quot;IP Address&quot; IN CIDR(&quot;192.168.0.0/16&quot;)</td>
</tr>
</tbody>
</table>

User:

- currentUser() - Filter on user attributes connected to the current (logged in) user. Attribute needs to be of type User.
- currentReporter() - Filter on user attributes connected to current reporter in custom fields. Attribute needs to be of type User.
- user(user1, user2, ...) - Filter on user attributes having the specified users

Group:

- group(group1, group2,...) - Filter on any object connected to a user within a specific group.
- user(user1, user2, ...) - Filter on any object connected to a user within a specific group.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>User in group(&quot;jira-users&quot;, &quot;jira-administrators&quot;)</td>
</tr>
<tr>
<td>User having user(&quot;admin&quot;)</td>
</tr>
<tr>
<td>Group having user(&quot;currentReporter()&quot;)</td>
</tr>
</tbody>
</table>

Project:

- currentProject() - Filter on any object connected to the current project in a custom field. Works only in the context of a ticket.

| Project = currentProject() |

References:

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
</table>
| outboundReferences (IQL, referenceTypes) | outR (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
    e.g. object having `inR(label = application, refType IN ('depends', 'installed', 'using'))`

Placeholders, references, and IQL can be combined in powerful ways. For example, you could add multiple object references to a custom field attached to an object, and then search those references for a specific key:

**Example**

```java
object HAVING inboundReferences(Key IN (${MyCustomField@0}))
```

Or a specific label, by using dot notation:

**Example**

```java
object HAVING inboundReferences(Label IN (${Portfolios.label@0}))
```

**Connected Tickets:**

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>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectedTickets()</td>
<td>All objects having tickets connected to them are returned.</td>
</tr>
</tbody>
</table>
**connectedTickets (JQL)**

Object having tickets connected to them that match given JQL query are returned.

**Notes**

- 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.

- To avoid confusion please remember that Insight Object Detail View presents *Unresolved* connected tickets by default.

**Examples**

- object HAVING connectedTickets(labels is not empty)
- object NOT HAVING connectedTickets(labels is empty)
- object HAVING connectedTickets()

**Object Types:**

- `objectTypeAndChildren (Name)` - Filter objects based on the object type specified by the Name and its children.
- `objectTypeAndChildren (ID)` - Filter objects based on the object type specified by the ID and its children.

**Example**

- objectType in objectTypeAndChildren("Object Type Name")

**More Examples:**

```
Owner = "Mathias Edblom"
Status IN ("Running", "Open") AND Group = "jira-developers"
"License Date" > "-30d"
objectType = "Computer System" AND Status = Running
objectTypeId in {1, 2, 3}
Name like "Micro"
"Operating System" = "Linux"
"System Date" < "1/Nov/18"
Created = "1/Nov/18"
Manager IS NOT EMPTY
"License Date" IS EMPTY
"System Owner" IN ("Mathias Edblom", "Tommy Nordahl")
"User Group" != jira-developers
anyAttribute = 123.123.123.123
"System Owner"."JIRA User" = currentReporter()
"System Owner".anyAttribute = 23
Host.Virtual = true
object HAVING inboundReferences("IP address" is empty)
object NOT HAVING outboundReferences()
object HAVING inboundReferences(object HAVING outboundReferences(Name = "Linux"))
object HAVING inR("", "Installed")
object HAVING outR("", refType IN ("Installed", "Uses"))
object HAVING inboundReferences(Name = "Linux", referenceType NOT IN ("Installed", "Uses"))
"System Owner" IN group("jira-users", "jira-administrators")
object not having connectedTickets(labels is not empty)
object having outR(Name Like "Microsoft" and objectType = "Operating System") and object not having inR("HotFix ID" = "KB2919355" and objectType = "Patch")
```

"connectedJiraIssues" is deprecated
**Objects containing quotes**

When searching for objects containing quotes, they need to be escaped.

Let's say you have an Object name "15" Screen, to search for it escape " with \, like \\

\textbf{Name= "15" Screen}\\

**Dot notation**

Dot notation is used to traverse down a reference chain.

To demonstrate. Given a structure with 2 object types

<table>
<thead>
<tr>
<th>Attribute</th>
<th>ObjectType: Employee Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Default string attribute</td>
</tr>
<tr>
<td>Using Software</td>
<td>Reference to object type Software</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>ObjectType: Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Default string attribute</td>
</tr>
<tr>
<td>Status</td>
<td>Status attribute</td>
</tr>
</tbody>
</table>

To query object of type Employee Details using a software with status attribute missing one would query

\textbf{"Using Software".Status is empty}\\

The following screenshots show the same example.
IQL - Order By

It is possible to add the order by suffix described by the following syntax

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.

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 no tcontain the attribute specified in the order by clause the order of the objects returned will be arbitrary.

The placeholder label can be used instead of the attribute name to order the objects by their configured label.

Example

```
order by label
order by Key desc
order by "My custom Attribute"
order by "My object reference"."Attribute on referenced object".asc
```

Read about where to use IQL

- Automation
- Insight Searching & Filtering
- Insight Custom Field
- Insight Referenced Custom Field