Esp@cenet levelx

# Espacenet Query Language (EQL)

Added by Roland Nelson, last edited by Vladimir Bataev on 06.03.2007 (view change)
Labels: eql

**ESP@CENET QUERY LANGUAGE**

**SPECIFICATION DRAFT**

## <CONTENTS>

## Goals

Build high-quality support for queries in esp@cenet levelx applications. This includes:

1. a well-defined and described esp@cenet query language (eql)
2. a common and well-understood grammar for building compilers to translate into 3rd-party search query languages
3. support for free-form user queries as well as multiple specific search flavours;
4. an implementation that contains configurable validation rules and smart compilation (using various heuristics)

| Tasks: thingsToDo | |
|---|---|
| ✓ (vb82170) Describe the esp@cenet query language (EQL) (del) | |
| ✓ (vb82170) Describe Java implementation of EQL. (del) | |
| ✓ (vb82170) Describe its compilers. (del) | |
| Add Task: Add | |

## EQL description

1. eql is a simple search query language that valid levelx queries are expressed in.
2. eql defines a set of scopes (tags) to group search terms into. During compilation into a target backend

language scoped queries are run against specific backend structures (zones, tables, etc).
3. search terms are case-sensitive, other
4. eql is used within levelx system to provide an abstraction above the target backend search query language and underneath user interface (which allows submission of either free-form queries or scoped queries only, both are translated into eql).
5. currently an eql statement is compiled into the target backend language for execution.
6. object representation of an eql statement allows easy validation of queries against a set of rules.

**EBNF for EQL**

```
<eqlStatement> ::= userLocale:<locale> and systemLocale:<locale> and dataSourceLocale:<locale> and
dataSourceName:<word> and <scopedQuery>[ and <scopedQuery>]+
<scopedQuery> ::= <scope>:<predicate>
   <scope>        ::= title|abstract|inventor|applicant|publicationDate|
                  accessionNumber|applicationNumber|publicationNumber|priorityNumber|
                  ecla|icai|ican|icci|iccn|preIpc
   <predicate>     ::= [not] <term>|[not] ([not] <term>[ and|or [not] <term>]+)
   <term>           ::= <basicTerm>|<quotedTerm>
   <quotedTerm>     ::= "<word>[ <word>]+"
   <basicTerm>       ::= <word>[<wildcard>]
   <wildcard>        ::= *|?|#
```

| TOKEN | ITS TRANSFORMATION | COMMENT |
|---|---|---|
| <LETTER> | ["a"-"z", "A"-"Z"] | |
| <DIGIT> | ["0"~"9"] | |
| <CHAR> | <LETTER> \| <DIGIT> | |
| <WORD> | <CHAR>+ | A word is just one character: a letter or a digit. |
| <LOCALE> | <LETTER><LETTER>_<LETTER><LETTER> | Locales are case-insensitive. |
| <QUESTION_MARK_WILDCARD> | "?" | |
| <HASH_WILDCARD> | "#" | |
| <ASTERISK_WILDCARD> | "*" | |
| <WILDCARD_SEQUENCE> | (QUESTION_MARK_WILDCARD)* \| (HASH_WILDCARD)* \| (ASTERISK_WILDCARD) \| (HASH_WILDCARD)+ (QUESTION_MARK_WILDCARD)+ \| (HASH_WILDCARD)+ (ASTERISK_WILDCARD) | |

Wildcards:

- number sign (exactly one character)
- ? question mark (zero characters or one)
- * asterisk (zero characters, one or many)

Aggregation symbols:

- Quotation marks "": all symbols within quo
- Parenthesis (): are used to impose rules of precedence different from default ones and change the order of evaluation.

### Available wildcards.

| ? | # | * |
|---|---|---|
| 0-1 char | 1 char (exactly one) | 0-max chars |

### Legal combinations of wildcards.

| 😁 | ? | # | * |
|---|---|---|---|
| **?** | ???? (0-4 chars) OK | ?# *(1-4 chars, see #?)* ILLEGAL | ???* (max chars, see *) ILLEGAL |
| **#** | ###? (3-4 chars) OK | ## (2 chars) OK | ###* (at least 3) OK |
| ***** | *? (max chars plus maybe one more, see *) ILLEGAL | *# (max+1 chars, see #*) ILLEGAL | *** (as many as possible, see *) ILLEGAL |

**Example queries in EQL submitted via web forms**

### Quick search flavour:

- database:x **and** (title:y **or** abstract:y)
- database:x **and** (inventor:y **or** applicant:y)

### Number search flavour:

- database:x **and** (accessionNumber:y **or** applicationNumber:y **or** publicationNumber:y **or** priorityNumber:y)

### Advanced search flavour:

database:x **and** (title:a **and** abstract:b **and** applicationNumber:c **and** publicationNumber:d **and** priorityNumber:e **and** publicationDate:f **and** applicant:g **and** inventor:h **and** ecla:k **and** icci:l **and** iccn:m **and** ican:n **and** icai:o **and** preIpc:F)

**EQL keywords**

| keyword |
|---|
| database |
| userLocale |
| systemLocale |
| dataSourceLocale |
| title |
| abstract |
| inventor |
| applicant |
| accessionNumber |
| applicationNumber |

| |
|---|
| publicationNumber |
| priorityNumber |
| publicationDate |
| ecla |
| ipc |
| ipcc |
| ipca |
| ipcci |
| ipccn |
| ipcai |
| ipcan |
| citation |

**EQL scopes**

At this point there are four distinct scope types:

## Text

Scopes: title, absract, claims, description, inventor, applicant (possibly others from other domains)
Implied operator: and

Open question: For a free-form query, what should be a default scope if the text attribute was not scoped explicitly?
Answer:matching free-form text to a set of modifiers has to be specified via esp@cenet application management (Vienna)

## Numbers

Scopes: application, publication, priority, accesion number
Implied operator: or

The common format for all numbers is [a-zA-Z]**[0-9]**[a-zA-Z][0-9].
Thus GB starts to compete with text scope during free-form query.
Detecting a number like 1000000 is easy.
Kind codes can also be detected.

(+ a special case)
There are nine countries/organisations which use another format, namely Australia, France, Gulf Council, Hungary, India, Italy, Mexico and the WIPO. In this case, the application number is made up of :

- a country code (two letters),
- the year of filing (four digits),
- "other" information (one or two characters),
- a numerical digit (five to six).

## Dates

Scopes: publication date
Implied operator: or
Formats:

- YYYYMMDD
- YYYYMM
- YYYY
- DD/MM/YYYY

Open question: How can we support date ranges?
Answer:date ranges are ideally to be entered as followed:

YYYYMMDD-YYYYMMDD
YYYYMM-YYYYMM
YYYY-YYYY
DD/MM/YYYY-DD/MM/YYYY

this also imposes the fact that the **/** format for date entry should also support the following case MM/YYYY

with

MM/YYYY-MM/YYYY as the date range search syntax

Open question: How about dates in free text format, like October 2007?
current answer:dates in free text format are **not** to be supported

## Classifications

Scopes: international and european classifications.
Implied operator: and
Formats:

**ECLA:**
The ECLA classification symbol is made up of a letter denoting an existing IPC symbol, followed by a number (two digits) denoting the section level (eg B65). Optionally, the classification can be followed by a sequence of a letter (eg B65D) denoting the subclass level, a number (variable, 1-3 digits, eg B65D81) denoting the group level, a forward slash "/" and a number (variable, 1-3 digits, eg B65D81/32B) denoting the subgroup or full classification.
If you want to retrieve all levels below the subgroup level, please use the * symbol to truncate your search. For example, if you enter B65D81/38*, you will retrieve an important number of results, including classes such as B65D81/38B4, L2, G2, H, C1, etc.
If you enter B65D81/38 in the ECLA field, you will retrieve a lot less results relating to that particular subgroup.

**IPC:**
The classification symbol is made up of a letter denoting the IPC section (eg A), followed by a number (two digits) denoting the IPC class (eg A63), still followed by a letter denoting the IPC subclass (eg A63B). A number (variable, 1-3 digits) denotes the IPC main group (eg A63B49), a forward slash "/", and a number (variable, 1-3 digits) denotes the IPC subgroup (eg A63B49/02).

Open question: What is this supposed to mean?

You should not use wildcards as the data is autoposted, meaning that each symbol is indexed at different levels. For example: B (section level), B65 (class level), B65D (subclass level), B65D81 (group level), B65D81/32B (full classification).
Put differently, A63B49 will search for all symbols under main group A63B49/00 (including the main group itself).

## LEGACY EQL LIMITATIONS

The old document on EQL (called K2 Queries + CSR, available in Lotus) contained the following limitations:

- No more than four tokens in the search form field (this can be discarded to allow queries of arbitrary length)
- No more than twenty tokens in total (same story, this can be dropped)
- Wildcards are only allowed as suffixes

## Implementation notes

For a typical query lifecycle, here is the simple sequence of actions:

1. A user submits the query to the system.
2. The system creates a SearchCommand object to transfer the query to the controllers.
1. The system creates a Query object and populates it with ScopedQuery.
2. The system applies validation rules to each individual ScopedQuery and to the Query itself.
3. The system calls toEql()/toString() that traverses the Query and its ScopedQueries and creates a string with a canonical EQL statement (EQL search query string).
4. The system passes the EQL statement to CSRA.
5. CSRA calls an injected implementation of an EqlCompiler that corresponds to the selected data backend.
6. An EqlCompiler compiles the EQL statement into a query string in the target language.
7. EqlValidator ? does semantic validation of the query in the target language (if needed).

Question:

- How the SearchCommand can be used in conjunction with free-form query?

Todo:

- Add Verity K2 EqlCompiler.
- Add Epoque EqlCompiler.
- Add Lucene EqlCompiler. not yet necessary. Optional

-----------------------------------------------------------------------------------------------
On free-form query implementation.
-----------------------------------------------------------------------------------------------

A free form plain text search query can be compiled into a set of EQL statements using common sense and various heuristics.

Todo:

- Specify free-form query translation procedure.

-----------------------------------------------------------------------------------------------
On legacy query validation rules (formerly part of bizlogic)

---------------------------------------------------------------------------------------------------

There is a configuration per database (but the one and only database is hardcoded into query translation action).
For each database various validation conditions (parsers and their aggregates -- parser methods) are configured per field.
If any of these conditions is not met, a separate factory produces a tailored exception and an error message.

[+] Verbose error handling.
Flexible but complex set up.

[-] Biz logic knows about specific database and its field (validation happens too early).
Validation is run against form fields.

First each search field´s content (subquery) is validated against the specified database config.
Then db parsers are run.
Then the final validation method is run.

Todo:
---------
Convert the legacy bizlogic validation rules into plaintext so they can be evaluated.
Code the necessary ones as ValidationRules and inject them into ScopedQueries.
See where all the leftovers go.

---------------------------------------------------------------------------------------------------
On difference in parsing the queries between level1 and level2 systems.
---------------------------------------------------------------------------------------------------

level1 EQL grammar does not contain various keywords like (an, ab, ti, etc) because it always parses individual subqueries independently and never together (with the only exception of IC symbols that can be stuffed into a single field, and it already hosts a constellation of quick hacks to make it work).

level2 eql grammar does contain those so that free form search is possible.

Todo:
---------
Merge the search criteria modifiers from level2 grammar with level1 grammar´s. This would become levelx EQL grammar.

---------------------------------------------------------------------------------------------------
On the usage of an remote service to enhance search queries.
---------------------------------------------------------------------------------------------------

This remote service is, in essence, a translation service that finds the translations of individual search terms and their combinations in other languages. It may also employ some variation of a soundex algorithm to find matches, but i guess this is the safest assumption that we can make.

Thus, the rationale behind using this service is to increase completeness of a search by adding multiple translations of the search terms (in the first term those should be translated into the language of the target datasource). This is helpful exactly in the cases when the data source contains (and has indexed) data in multiple languages or when the user/system locales do not match the locale of the target data source.

My opinion is that a levelx system should know as little as possible about this external translation service. There should only be a deployment-time switch that either enables or disables the whole communication between systems completely.

Thus all configuration and support of this particular service should be pushed over to its provider.To facilitate this two things can be done:

1. EQL can be extended with two new keywords to provide the service with additional information:
userLocale -- to denote the current locale of the user interface,
systemLocale -- (formerly source/hostname) to denote, in the first place, the country/national office where a levelx system was deployed.
If these two match, then most likely we can guess the language that the user is conducting the search in.

2. A running levelx system should periodically poll the service for changes to find out whether new locales or scopes are supported and cache this configuration data. If the service fails to respond then the levelx system should temporarily stop further remote requests. However, the system can be started with a sensible default set of supported scopes to reduce the amount of network communication.

Most likely the external service will "enhance" the query on per-scope basis. Thus the remote service can be invoked once all ScopedQueries are validated, and the Query itself if marked as valid, but before each ScopedQuery is serialized into EQL. Hence, scoped queries provide added benefit since there is no need for parameterizing queries, as proposed earlier.

Thus it makes little sense to contact the service once an EQL statement has been compiled into the backend query language, unless there is some "enhancement" that requires knowledge of the target language (this is not specified yet).

Question:
-------------
All these manipulations with languages might require clearer understanding of how we use locales. We clearly recognize the locale of the user interface (we are more concerned with its language), so that we can localize the user inteface. We also recognize the locale (this time, the country) of a deployed system and use it for multiple purposes: from correct navigation and url-rewriting to setting sensible default values for user locales. However, a third variable in this equation, the datasource, is a locale-less entity that is only distinguished by its name. What is even worse, during rpm package build default datasource files are overwritten with localized (in this case, tailored to the specific country requirements) datasource files. So there is definitely some room for improvement.

Question:
-------------
How highlighting of the search terms can be added to the displayed results?

## Children

Hide Children | View in hierarchy | Add Child Page

Coping with EQL injection (Esp@cenet levelx)

even though this first specification is to be within a 100% esp@cenet context, we should foresee the specification becoming broader in its scope and implementation when adding REGISTER (epoline) modifiers

e.g. ATTORNEY, REPRESENTATIVE ....

(the abbreviation eQL is future-proof, as we internally will only have to rename it to the epo-Query-Language... 🙂

Posted by Roland Nelson at 26.02.2007 15:37

Hej,
there is a RC1 of the EQL JavaCC grammar attached to the page.

Posted by Vladimir Bataev at 27.02.2007 15:02

Followed by RC2.

Posted by Vladimir Bataev at 27.02.2007 16:22

please elaborate a little more on what is unclear regarding the IPC Classifications (**Open question: What is this supposed to mean?**)

Posted by Roland Nelson at 27.02.2007 17:05