Kin Query Language

Query Language Definition

Queries (data request)

Queries are based on two text strings, for Standard Query Type. First string defines query criteria (qReq - Query Request), the second the data we want in the results (rReq - Result Requirements).

a) qReq is a comma separated list of conditions to be searched.

*Conditions are:

- Equality (= or ==), Different (<> or !=). For text strings, = means partial match (i.e. "is into"), and == means perfect match (case sensitiveness depends on modifiers)
- Greater (>), Smaller (<), Greater or equal (>=), Smaller or equal (<=)

*Fields are defined by their fully qualified name, or a partial name referring to a previously named parent. For an instance, it is the same "Worker.Age=[25..30],Worker.Type=8" and "Worker.Age=[25..30],.Type=8" (note that second field is specified by reference to previous object name, just preceding it by the dot, meaning "the same path than before"). This is a convenient way to save some space and typing, as usually you refer several times (or all) to the same object, different fields. Note that by default, starting by dot means "use all the path", so it will try first to substitute the last field name for the new one. If this fails (no field by that name), it will extract tree elements, one by one, from the end, till the field exists (i.e. will scan up in the tree, to its parent, and so on). Thus, "HHRR.Worker.Salary>50K,.Desk=3" will try to match HHRR.Worker.Desk to 3, but if that field doesn't exist, will try next with HHRR.Desk (obviously, if that also fails, there's no more tree to navigate, so then will fail with an error "field not defined").

An array field may be specified with empty brackets, meaning you want any of its elements to be taken into account (i.e. any of its elements may match), or an index between brackets, meaning to compare only that item.

*Constants to be compared, are the same type (or a convertible one) than the field specified. Might be:

Numbers:

Number notation:

-Floating point: 23.05 or 2.305E1 or 2.305E+1 or +2.305E+1

-Integers: -25 or +25 or 25

You may use multipliers for short: 25K4 for 25400, 25M1 for 25100000

Expressions:

-One unique value (12345)

-A list of values ([1,2,3,4])

-A simple range ([12345..67890])

-A list of ranges ([1..5,11..15,21..25])

Text Strings:

Just enclosed between double quotes (“), and escaping the quotes themselves (if present into the text) with a backslash.

Dates:

There are different types for dates. For unix style timestamps, a 32b numeric value may be used. In general, also a full date value may be expressed as YYYYMMDDHHMMSS, but there are other shorter or longer representations:

-Unix style: 1044290765 (seconds since 1/1/1970)

-Standard datetime: 20040815180959 (18:09:59 Aug 15th, 2004)

-Standard date: 20040815

-Standard time: 180959

As there's automatic conversion between compatible types, you may use any convenient representation for all of the date/time/datetime field formats. But to make clear what you mean, when you use a different type of the field type, please use type specification letter, like u1044290765, d20040815, t180959. No need for the Standard DateTime, as it's long enough to be automatically detected.

Expressions:

-One unique value (d20040815)

-A list of values (d20040808,d20040815)

-A simple range ([u1044290765..u1044377165])

-A list of ranges ([d20040816..d20040819,d20040822..d20040836])

Note that automatic conversion for ranges and not-equality comparisons works depending on the size of the field: if given number has to be expanded (e.g. a date is supplied, and a

datetime is needed), the value is saturated (i.e. time is set to 23:59:59) when value is end of a range or a < or <= comparison (top value); or truncated (time set to 0:00:00) for the beginning of a range or a > or >= comparison.

Geographic data:

Always enclosed in parenthesis, first two values are coordinates (latitude, longitude), then may be one or two values more. Last value is always a distance in meters (resolution up to 0.5m). When there's only three values, distance is a circunference radius that matches all the items within that cilinder (i.e. no matter the height value might be, on G3D items). When specified four values, third one is height of a sphere center, distance is sphere's radius, and only match points laying into that sphere. Specifying four points and a G2D target results in an error.

*Notes:

1. List of conditions is like an AND clause (all of the conditions must be met to get a match); lists of values or ranges is like an OR (any of the values matching). By now, OR for the field list is not implemented, as it can be issued as more than one queries, and although system cost is usually bigger (there's jobs to do several times instead of one), applications seldom need such a condition.

b) rReq is a comma separated list of records and fields to be retrieved for the matching set.

*Format is similar to the field definitions of the request, and the same rules apply for shortening path names.

*Depending on the field type, it is returned:

-For a straight field (a field that is a simple value): the value

This applies for numbers, basically.

-For a referencing field:

Depends on the referenced type:

-Text string: the value is always returned. If request type is Standard Text Request (unpacked), also the reference is present at the main record, so client can link it to the value that receives separately.

-Relational Record (e.g. reference to another record object): by default (no modifiers), the whole record is retrieved and returned (as usually is nonsense to request record ID, an internal value not useful for applications). If, for some reason,

you want the reference (numeric ID to the record), just ask for that field reference ID, like "Worker.Boss.ID"

-For a subrecord field:

This is an artificial field (no real field), all its fields are returned instead (i.e. all the fields with a name starting by the one specified plus dot).

-For arrays:

Depends on what you specify: empty brackets mean all the array; a bracketed index means only that element, as a scalar (i.e. as if was not an array but a single item).

Save (store) requests

Appart from searching, you have to feed the system to contain data. For storing information into data base system, you use Save Requests.

A Save Request consists on a series of objects and fields definitions with assigned values. The first object named, is considered the target of the operation, and thus all other objects (if present) must be referred from the target one by some field. You may save a whole record, or just some of its fields, even a single bit. The rest remain unchanged (if record existed) or set to default values (if is a new record).

Request is directed from the QM to the corresponding Manager, being a simple IM, or better, a TM that handles all the process with more intelligence. For an instance, when you create a new record (row), defaults are 0 or empty for direct IM save, but should be set by an associated TM (that you should have written for that objects).

A typical Save Request might be something like:

```
VisV.ID=0,.Dele=300,.Vis=20040817113000,.Inm=43506,.Cli.ID=0,.Cli.Nom=”David”,.Cli.Cog[0]=”López”,.Cli.Tit=1
```

Some hints:

- Target object: VisV, Referenced objects: Cli (by VisV), Noms (by VisV.Cli.Nom), Cognoms (by VisV.Cli.Cog1)
- We are requesting the creation of two record objects: one in object VisV, and another in object Cli. This is explicit as we say .ID=0 thus asking for a new one. In Noms and Cognoms objects, we are just requesting a new value to be set (or reused if exists), by assigning its reference fields .Nom and .Cog[0]. All other fields are to be left with default values.
- For this object, we have a TM that gets the request (instead of VisV and Cli IMs), does some checks -like checking that there's a record with ID=43506 in Pis object (.Inm is a reference to Pis object), also that Vis date is in future, and that a tag exists on the query with logged user data-, creates and sends a Save

Request for Cli (directly to IM, this time), retrieves result, and if succeeded, assigns VisV.Cli to received ID and creates a Save Request for VisV (direct to VisV IM), retrieves result and sends final request result to CAP.

- IMs are responsible for automatic fields, like creation date, modification date, and also user who creates and/or modifies (included in an extra tag attached onto the queries).