A Geophysical Resource Web Services (GRWS) query is the search for resources at a repository by a user-agent, employing a filter, as part of a transaction, possibly accompanied by a set of instructions.

Resource queries against the GRWS repository will take place in a well-documented and orchestrated workflow with multiple transaction junctions. Below is a description of those junctions.

a :: GRWS_SubmitQuery.jar is invoked, with a combination of parameters to denote resource type of interest, spatial context, temporal constraints and optional nominal constraints. User input is minimally quality-checked for sanity, likely query execution time and delivery mechanism.

b :: GRWS_SubmitQuery.jar serializes an appropriate OGC Filter Encoding XML fragment, composes a GRWS_Query SOAP message and submits the request to the GRWS_Query (GRWS_Query.wsdl) web service. The “submit query” GRWS role may be instantiated by any agency, in any modern programming language.

c :: GRWS_Query receives the resource request and performs extensive quality-checking on the request.

d :: GRWS_Query performs authentication (if necessary) and creates “query” session, or ticket, in database for tracking purposes. Successful negotiation would result in an GRWS ticket (e.g. identifier linked to a session|transaction, tagged with an expiration date, etc).

e :: GRWS_Query dispatches query to threaded handle$RESOURCE$.java class and notes in database, for given GRWS ticket, that query was received, and dispatched for “handling”.
handle$RESOURCE$.jar receives the query (in OGC FE format) and builds a Hibernate query targeting appropriate database elements and constraints. $RESOURCE$ shall be replaced with a string particular to a given resource type. There may be many different types of resources, but only one resource handler agent for a given resource type. The objective of the resource handler is to perform workflow steps specific to it’s specialty, within a larger, scalable workflow.

handle$RESOURCE$.jar issues the Hibernate query against the GRWS repository.

handle$RESOURCE$.jar receives the query Hibernate query response and serializes XML content appropriate for it’s particular resource specialty.

handle$RESOURCE$.jar returns the query results as XML in the given resource format (ex. positionCollection) to GRWS_Query.

GRWS_Query receives “expected” results (as XML resources), performs minimal quality-checks and prepares a SOAP message in response to the client.

GRWS_Query returns a SOAP message to the client, with either an attached payload or an “in body” payload, depending on the requested delivery mechanism.

GRWS_SubmitQuery.jar receives the SOAP message from GRWS_Query and takes appropriate action as requested by the user. For example, print query results to standard output.

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

Optional Proxy Pre-Workflow

.1. A basic HTTP GET-enabled client is invoked, with a combination of parameters to denote resource type of interest, spatial context, temporal constraints and optional nominal constraints. User input is minimally quality-checked for sanity, likely query execution time.

.2. Basic HTTP GET-enabled client composes HTTP GET url and
issues GET request to appropriate GRWS Query proxy on another server.

.3. GRWS Query proxy receives request, through standard CGI name=value parameters, minimally quality-checks the request conditions and translates the request into a SOAP message appropriate for the GRWS Query web service.

(rejoin workflow stage (a) above and continue until stage (l) above)

.4. GRWS Query proxy prepares matching XML resource results for return to the caller. Included in this step of the workflow is the likelihood of incorporating XSLT filters to translate resource XML into flat ASCII text, or other (as needed) formats.

.5. GRWS Query proxy prints matching results to standard output back to the HTTP GET-enabled client – conforming to standard CGI constructs.

.6. Basic HTTP GET-enabled client receives results through standard input, conforming to typical CGI behavior, and returns results (or performs actions) requested by the user.