RDMS Client implementation – Steps

This memo (information paper) describes the steps to be taken for implementing (web service) clients for the so called RIS Data Management Services – Application.

The RDMS service consist of a GUI (1), where authorized users can specify mutation and download data etc. For applications a so called web service interface (2) is available (see figure below).

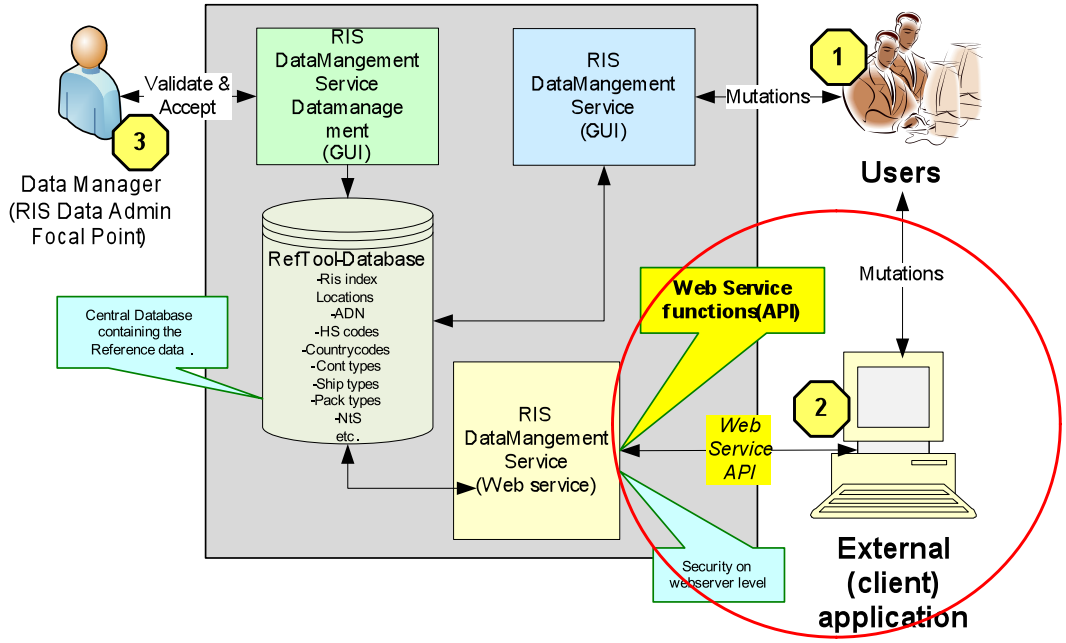

Figure 1, RIS Data Management Service context

There are 2 versions of the RDMS application available:

1. The so called Production version: that is the version with the actual Reference data and the one that will be used for the daily Reference data management.
2. A so called TEST version: that is the version and environment which must be used when developing new clients and for testing and demo's. This version will contain test reference data, which is not the same as the actual reference data. Connection to the Production version will be allowed after finishing some tests in the TEST environment.

The applications are available at the following URL's:

TEST RDMS:

GUI: http://test.risdatamanagement.ris.eu
WSDL: http://test.risdatamanagement.ris.eu/refws/v1/RefWebService.wsdl

Production RDMS:

GUI: http://risdatamanagement.ris.eu
WSDL: http://risdatamanagement.ris.eu/refws/v1/RefWebService.wsdl

For accessing, the application individual accounts are necessary and can be requested at:

ERI-datamanager@risexpertgroups.org

Codes maintained by the RIS Data Management Service are:

- ***RIS index***: describing several RIS objects on waterways (such as junctions, locks, bridges, berths, gauges etc).
- ***ADN codes****:* as specified in 2008/68/EC
- ***HS codes****:* Harmonised System codes from the Customs organisation (non-dangerous goods).
- ***ERI Locations****:* (also known as SRS codes) as used in ERI, electronic reporting using the ERINOT 1.2 message. At the moment these codes have an overlap with the RIS index, but this ERI location set also includes all the international locations outside of Europe.
- ***Container Types****:* to identify the type of container (ISO 6364).
- ***Country codes*:** the several countrycodes for all the countries (ISO 3166).
- ***Inner Package type:*** (UN Rec 21).
- ***Shiptype:*** *to describe the type of transport* (UN Rec 28).
- ***Notices to Skippers*** *codes (27 types, only publication).*

The available Web service functions for the RIS Data Management Service are:

Function	Description
matchByCode()	Find information based on reference code
matchByName()	Find information based on the name or description of the reference data.
getMutations()	Request all changes (records) in a given period for a specific reference data.
getDataXML()	Request all the current records of a particular reference data type (master dump of the reference data).
getRisDataXML()	Request specific RIS Index records (possibly a sub selection). For example by country or by function.
mutateDataXML()	Mutate specific (1 record) reference data (add new one or change existing). These mutations will be processed immediately without human interaction. However this function is only possible for the country specific data (such as locations) and only the country specific records for the specified user, other mutations will be rejected.
requestMutationXML()	Request a mutation (1 record) for specific reference data (add new one or change existing). These mutations will always be validated, in accordance with the procedures (manually) by the Data manager (against the applicable International standard) before publication (the data manager can reject or accept the request).

Implementation steps for new RDMS clients:

Below the most important steps for implementing new RDMS (web service) clients are described:

1. Request (collect) the RDMS documentation and specifications, consisting of (at least):
    - Maintenance procedures reference data and ris index Annex 3 to the report "consolidation of the RIS index and reference data" - 22 August 2010
    - Functional SPECIFICATION REFERENCE data Management tool data Annex 4 to the report "consolidation of the RIS Index and Reference data" - 6 September 2010
    - RIS Data Management Service API Service - This document describes the webservice (API) Interface functions of the so called RIS Data Management Service - 7 December 2010
2. Request an account for the TEST RMDS application.
   An account is needed for accessing the application and using the MutateDataXML() en RequestMutationXML() functions. With the account it is also possible to use the GUI to fill in manual requests, check requests and search for reference data.
3. Implement the NEW client using the TEST RDMS application (see TEST URL's).
4. Perform internal tests using the TEST RDMS.
    - Test if all the used web service functions in the client are working (especially the GetDataXML, and Mutate functions).
    - Validate the results using the RDMS GUI (look if the data is available etc).
5. Perform some tests in cooperation with the RIS Admin focal point / Data manager and or RDMS application manager.
6. After validating the results (RIS Admin focal point) in the TEST RDMS, the NEW client will be allowed access to the Production RDMS.
7. A new account will be created in the Production environment, after which the client can use the Production RDMS (switch URL's).

+++

Implementation of a Client (example)

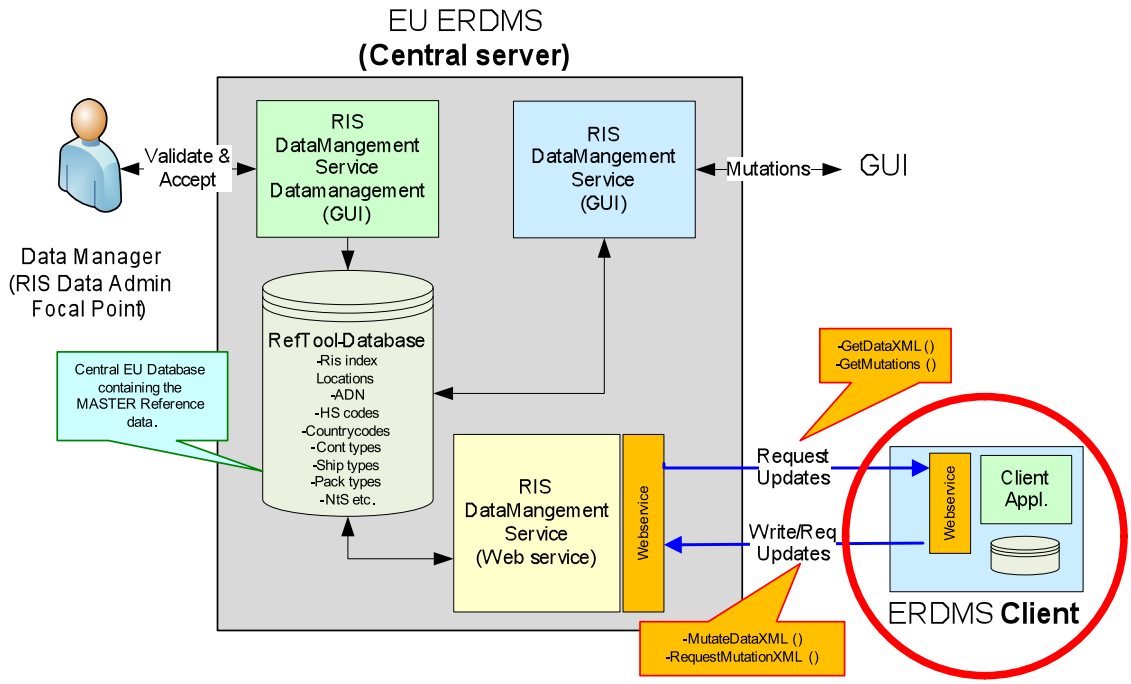

Figure 2: ERDMS Client context

Implementing a ERDMS client, which must be in sync with the EU ERDMS database, can be based on the Web service interface, using the following functions:

Request (data) Updates:

- Get DataXML() to update/fill whole tables (initially) on the client side.
- GetMutations() to get the last mutations since the "Last GetMutations" time up to now from the central server.

Write/ Request Updates (mutations):

- MutateDataXML() to upload locally accepted mutations to the central server (only country specific data).
- RequestMutationXML() to upload a mutation (change) request to the central server (requires validation of the EU Data manager).

Client Functionality (requirements) to implement in a ERDMS Client

Request data updates (client <= server):

- On the client side it must be possible for the local Admin to initially fill (sync) the tables with the data from the ERDMS (EU server).

  Actions:
  - Select table or tables on the client to sync.
  - Empty selected table (actual and history records).
  - Use **GetDataXML()** to get all the actual records of the selected table(s) from the central server and store those records in the client database.

- The client must periodically (daily = configurable), automatically update his database, by requesting all the mutations (changes) from the server and update the client database accordingly.

  Actions:
  - Implement a *time* or *manual* trigger functionality to update / sync the client database periodically.
  - Use **GetMutation()** to get all the updates since the last time up to now from the central server (no "last since" means 1-1-1970 = all).
  - Register the current time (now), this will be the "last since" time for the next update.
  - Process all the updates (result of GetMutations() function).
    + be ware that mutations can also contains deletions.
    + use the original (record) version nr of the central server (don't increment record version nr).
    + existing records (based on refcode) must be replaced (independent of the record version nr)
    + none existing (new) records must be added.

Write data updates (client => server):

- After accepting a mutation on the client by the local Admin, the client must mark the (accepted) mutations, so it can be send immediately to the Central server (update server).

  For country specific data (RIS index, Locations) the update must be uploaded using MutateDataXML(). For all other data types RequestMutationXML() must be used.

  Actions:
  - Use MutateDataXML() after accepting a mutation on the client to send (RIS index & Location) mutations to the central server.

    MutateDataXML (country specific data):
    After accept on client side, mark the accepted mutation and use MutateDataXML() to send all the marked mutation(s) to the central server.
    Only after uploading the (marked) mutation successfully the mutated record will become actual (definitive) on the client (in between it will have an pending state).

+ If MutateDataXML call is accepted (returns successfully) on the central server only then the mutation will become definitive on the client side:
  - increase version nr of accepted mutation on the client.
  - make accepted mutation definitive on the client (replace existing record in the client database if applicable).
+ If MutateDataXML call is rejected (i.e. rejection, because of a wrong version nr or invalid/incomplete data etc by the central server) then:
  - give warning/signal to local data manager on the client side about the rejected record (can be done by putting it on the request/reject list again on the client side including putting the rejection error in the remarks). Be aware that the mutation is not accepted on the client side, so mutation will not become definitive/actual.
  - Finally log this (reject) error situation (client side).
+ If MutateDataXML call fails (system failures, like cannot connect to the server etc) then:
  - keep on trying to send (marked) mutation(s) to central server (every hour for a certain period => configurable).
  - if finally the mutation cannot be send to the server then apply the same action as specified for a MutateDataXML rejection (give warning to local datamanager by putting it on the reject list etc).
  - Finally log this (system) error situation (client side).

RequestMutationXML (not country specific data):

After accept on the client side, mark mutation and use RequestMutationXML to send mutation request to the central server.

The client will not get any notification when the requests are accepted or rejected by the EU datamanager on the central server. Accepted (requested) mutations on the server will be received as updates (via GetMutations()). Rejections must be handled manually. An additional informative list(maintained by the local admin) can be used to inform the local Admin about all the requested mutations.

+ If RequestMutationXML call is accepted (result=ok) then the mutation will be put on a requested list (additional informative list for the local admin).
  The requested and accepted Mutations will never become definitive (published) on the client side, but they will be synchronised (updated) automatically from the server (by GetMutation calls)).
+ If RequestMutationXML call is rejected (wrong version nr, incomplete data etc) then see MutateDataXML rejection situation.
+ If RequestMutationXML call fails (system failures like cannot connect to the server etc) then see MutateDataXML system failure situation.
+ If RequestMutationXML record is rejected by the EU datamanager (after the call itself succeeded) then the restore actions will/must be done manually (EU datamanager will inform requester etc).

+++