Specifications FADA import tool

Version 14/10/2015 - V1.1 Draft 3.1 - Pre-final version

Authors: Aaike De Wever, Michel Kapel

This document is used for launching a call for offers for developing the FADA import tool. This document is intended as a guidance for constructing the web application, but as it consists of a complex database and the reconciliation of the needs from 'biologist' users and the technical needs and possibilities, it is likely that specific requirements will have to be clarified and refined along the way. This version includes changes -marked in red- after the work of this tool was initiated.

Supporting material for this specification document can be consulted at https://github.com/aaikedw/fada-import-specs.

1. General background information

1.1 Project status

The Freshwater Animal Diversity Assessment (FADA) database was constructed in 2009 following the publication of a special issue of the scientific journal *Hydrobiologia* in which taxonomic experts described the biodiversity of around 60 organism groups. This work was funded by Belspo and supported by the Belgian Biodiversity Platform.

The FADA initiative and its networking activities involving taxonomic experts led to a follow-up initiative in the form of the EU FP7 project BioFresh (Biodiversity of Freshwater Ecosystems: Status, Trends, Pressures, and Conservation Priorities; www.freshwaterbiodiversity.eu/), for which RBINS was in charge of constructing a data portal. This project ended officially in April 2014, but we have recently (Spring 2015) finalised the last technical updates (including the implementation of a 'Data Portal Import Tool' – DPIT).

Since May 2014, we focus on the FADA database again through the BRAIN AquaRES (Aquatic Register Exchange and Services) project – odnature.naturalsciences.be/aquares. We aim to improve the FADA database, by both

streamlining the database import procedures and setting up data exchange with the World Register of Marine Species (WoRMS) and the Register of Antarctic Marine Species (RAMS). In order to implement this project, we are looking to build a web application “FADA import tool” for importing data into the FADA database.

1.2 Database status

Currently (18/11/2014), taxonomic checklists for 16 organism groups are published on-line. In total these checklists contain around 47.000 names and thus represent almost 1/3 of the number of accepted names for freshwater animal species, which is estimated at roughly 150.000 species.

1.3 Organisation in groups and volumetry

The organisation by organism group, corresponds to the initial organisation as was adopted for the paper publication. In fact, these groups represent an “operational unit”[1] for which a taxonomic editor was found to produce an informed estimate of the number of known species. These organism groups may represent different taxonomic levels, e.g. classes, orders or families.

For database purposes and/or because no taxonomic editor is found to provide a checklist for the entire group, we may subdivide the task and create a new (sub)group. While taxonomically this represents the creation of a real subgroup, we do not aim to introduce any hierarchical structure at the level of the groups themselves, as these remain “operational units” in the first place.

In terms of size, the checklist for the different groups vary from not even 100 species names to around 16.000 names for Vertebrates-Fish. Unless, at some stage we will further extend the scope at the database to non-animal freshwater groups (other than the “Macrophytes” group, which is currently included), which is unlikely to happen within the duration of the AquaRES project, the total number of (accepted) species names is unlikely surpass 200.000 and 16.000 for individual groups in the near future.

Note: As DwC-A files may include non-freshwater and fossil species (which have to be filtered out at some stage), the number of records in these files can be higher.

2. Technical background

2.1 Technologies used for the BioFresh and FADA database and web applications

The FADA database is managed in PostgreSQL together with the BioFresh occurrence database. The FADA website and the original import scripts were developed in Ruby. The BioFresh portal was developed in the Groovy programming language.

2.2 Existing data import tools

For the BioFresh database, we developed an import tool for occurrence data in collaboration with an external developer (Sylvain Renaudier). In essence, the interface for the FADA import tool will be quite similar to that of the 'Data Portal Import Tool' (DPIT). See screenshots 1-3 below.

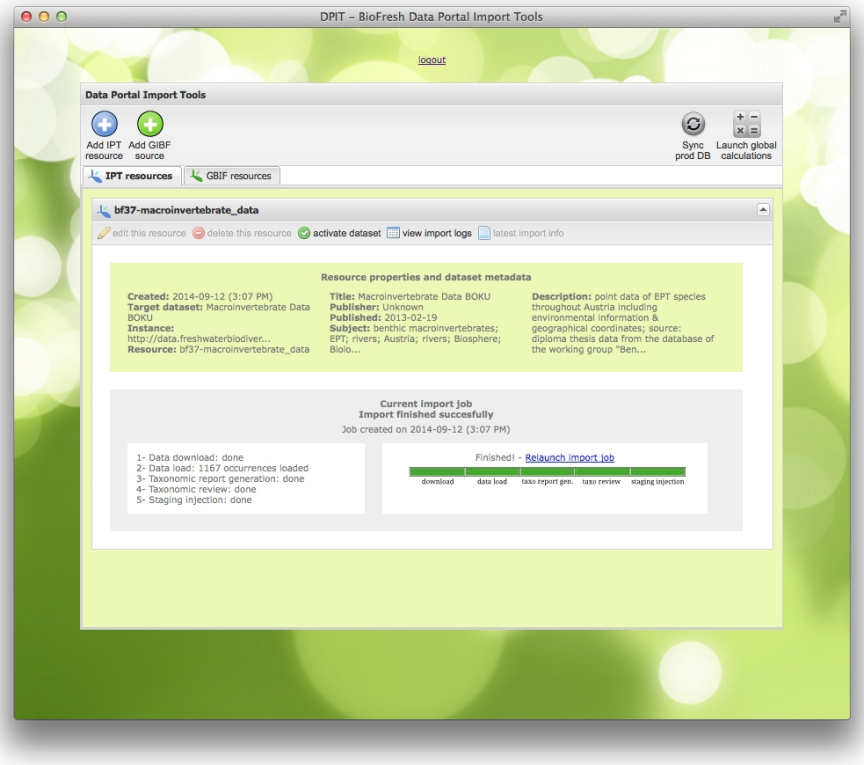

Fig. 1: General user interface of the DPIT.

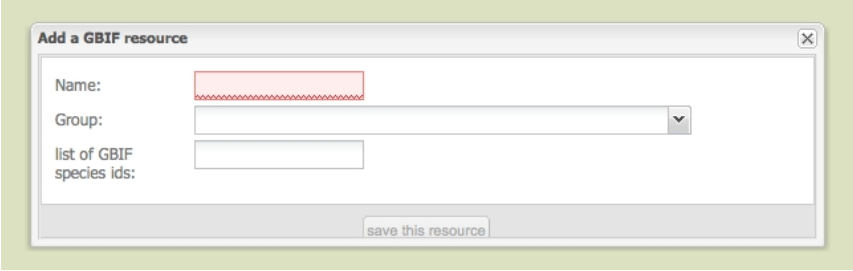

Fig. 2: Resource creation dialog of the DPIT.

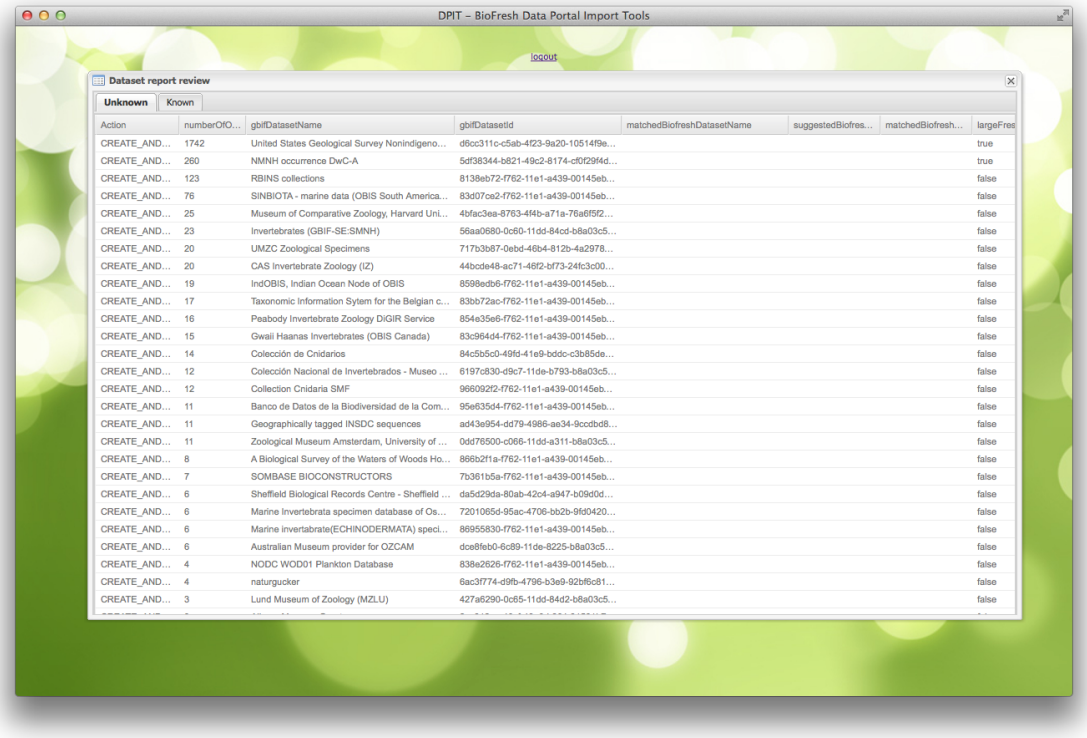

Fig. 3: Example data validation interface of the DPIT.

For importing data into the FADA database, original Excel-file import scripts were developed in Ruby. With the exception of the scripts for injecting the data into the database tables, the FADA import scripts were re-written in Groovy. Procedures and scripts for importing Darwin Core-Archive (DwC-A) data in FADA have to be developed from scratch.

2.3 FADA database structure and planned changes

SQL scripts current database structure

The definitions of the tables and views in the fada schema are included in the ./database-info folder. For information this folder also contains the definitions for the "importandsyncfada" schema as used for processing Excel imports. Note that this is work in progress, but I included this as it may provide inspiration on how to organise a staging area for processing the import files.

Description of the main database tables and background with regards to the database structure

The FADA database structure is described at the BioFresh-wiki. The species register (also registry) is described at http://trac.bebif.be/wiki/BioFresh/DatabaseStructure under the heading "Organisation of the central species registry".

The original database structure was elaborated to accommodate a hierarchical browsing tool through the FADA web application (see FADA Taxonomic Tree browser). Imports were seen as a one-shot thing, where updates would be preceded by a complete deletion of the earlier data, without the need for permanent identifiers. Through BioFresh and exchange projects, the use of the database was widened, and there was a need to introduce more permanent identifiers in order to link the FADA database to the BioFresh species register (hence the *biofresh_key* tables), and to facilitate the export of the hierarchical data into a flat, denormalised format (hence the *genus_to_family* table and the *fst_for_genus* and *fst_for_tribe* views).

While the organisation of the database may be improved in certain areas, we should maintain the compatibility with the original FADA website fada.biodiversity.be/, as we currently have no plans to rework this application. E.g. while the tables with regards to "observations" are barely used, these correspond to an option in the FADA interface, which we don't want to break at the moment, and thus should leave those tables untouched.

basically related to the organisation of the *biofresh_key*-tables and the flat taxon tables. The "biofresh keys" could be integrated in the taxon, species and synonym table eliminating the need for specific tables (this would require a small change in the portal app and export scripts). Instead of having a combination of the the *genus_to_family* table and the *fst_for_genus* and *fst_for_tribe* views, we could opt for extending the *taxons* table with a flat version of the complete taxonomic hierarchy (thus adding the fields: kingdom, class, order, family,… genus, specificEpithet, infraspecificEpithet).

As will be elaborated in the ./FADA-database-changes.md document, we currently (10/08/2015) envisage 2 main types of database changes: 1) changes directly linked to the FADA import tool and 2) changes linked to improving the overall database structure and performance. The organisation of the biofresh_key-tables and organisation of the taxon information is considered under (2 - chapter 4 in the database changes document) which should be dealt with under a separate contract.

Note: On 30/07/2015 Sylvain Renaudier notified me that the keys we are using to link the tables are currently not actual "foreign keys", which means they are not indexed. This issue can be addressed by adding constraints for the ID fields. Michel will need to test whether this has an impact on the FADA-app, if not, this change could still be applied independent of (2).

Planned and needed changes in terms of database structure

In addition to expected changes due to the implementation of a FADA-import tool with its own progress and log database tables, we foresee the need for specific changes based on the required fields to implement the data exchange using the DwC-A exchange format (see 4.1) and certain requests from our editors (categories for aquatic/water dependent species). These changes include the need for the following fields/tables;

- A record level "modified" timestamp, this could be coupled to the import log timestamp
- The isFreshwater (default yes for FADA datasets), isMarine, isTerrestrial, isBrackish fields/flags (also to be added to the xls-template) and "Freshwater aquatic/water dependent" category and subcategory (in xls-template, but currently not stored in the database itself (other than in the import "distr_table")). This could be either implemented in the species table or a separate table (cfr. the regions table).
- Need to conserve "provider IDs" for taxons and species names need to be considered. Can this be done by updating the "biofresh key"-tables or should we work out another solution?
- Extend groups table with selected metadata fields from DwC-A EML including; alternateIdentifier for datasetID/URL, keywords and keywordThesaurus, intellectualRights and bibliographicCitation for the dataset as a whole and the URL/location of full EML metadata.

In addition to the required changes described above, we identified a number of issues that merit to be considered to facilitate future developments;

- While the species table currently contains a “status_id” field and the entries in the synonyms table are by definition all invalid, subjective synonyms, this is not a straightforward solution to provide the “taxonomicStatus” in the DwC-A-export (esp. for providing the “original combination” or “objective synonym”/“basionym” or “homotypic synonym” - terms respectively used in zoology/botany). A solution for this issue should be considered while addressing the FADA database structure. The main question will be whether this issue can be solved without breaking the backward compatibility with the FADA app.
- Bibliographic references are currently organised in 2 tables, *greferences* for references pertaining to a group and *publications* linked to individual species (and Rotifera references to be imported are still in a separate schema). Although joining these tables at this stage may not be an option due to the need for backward compatibility, their integration will likely be considered in the future.

2.4 “biofresh keys” as persistent identifiers and the link between the FADA database and the BioFresh species register

As mentioned under 2.3, the “biofresh key” tables originated from the need to have a (more) permanent identifier to establish a link between the FADA database and the BioFresh species register. At this stage there are 3 tables; *biofresh_key_species*, *biofresh_key_synonyms*, *biofresh_key_taxons*, each consisting of an id (considered as “biofresh key”), the id in the respective table for species, synonyms or taxons and the scientific name and/or other crucial data to establish the link with the original tables. Currently, during each update, the ids for the original tables are removed and the new ids are re-established through name-matching. In case of changes in the spelling of the scientific name or changes in the (parentheses of) authorship, this field is updated in the *biofresh_key* table.

If we design the import tools so that they only update the changed records, we could eliminate the need to get rid of the deletion of the original ids. By integrating the “biofresh keys” in the *species*, *taxons* and *synonyms* table as discussed under 2.3, we would further eliminate the need to update those tables separately.

The link between the BioFresh species register and the FADA schema are covered by the “biofreshdatasources” table (see definition in ./database-info/register-tables) where register.biofreshspeciesregistry.id = register.biofreshdatasources.fkbiofreshid

AND register.biofreshdatasources.intdatasourcekey = fada.biofresh_key[_species].id. These links remain stable, except if a species name is synonymized and the link to the fada.biofresh_key_species.id needs to be replaced by the fada.biofresh_key_synonym.id.

For new species names added to the FADA database, their presence in the BioFresh species register needs to be checked (as this register can be populated through other sources) based on the group and species name matching (both exact and phonetic). See further details under 6.2.

3. Overall FADA import tool specifications

3.1 Envisaged components of the FADA import tool

Data for import into the FADA database will either be available in one of the following formats, which each represent a different *data import module* (corresponding to different “tabs” in the main interface);

1. Data import interface for organism group-data provided in DwC-A format (for externally managed datasets, in particular those coming from WoRMS)
2. Excel template data entry and update interface for FADA checklists (default option for FADA editors)

Note however that, while data for a group is either present in the database as a “DwC-A” or an “Excel” resource, at some stage we may choose to “throw out” the data imported through an xls-file and import a checklist for a group in DwC-A (if this would be a more complete/trustworthy checklist for example), so there should be a mechanism to delete the data associated with a Excel/DwC-A resource and replace it by a DwC-A/Excel one (could be just creation of a new resource).

These two modules need to be integrated in a joint interface with the following *shared components*:

1. Resource metadata entry and edit module
2. Functions for synchronising the data imported into the FADA database/schema with the BioFresh species register table
3. Synchronise the staging database with the production database

3.2 General user interface

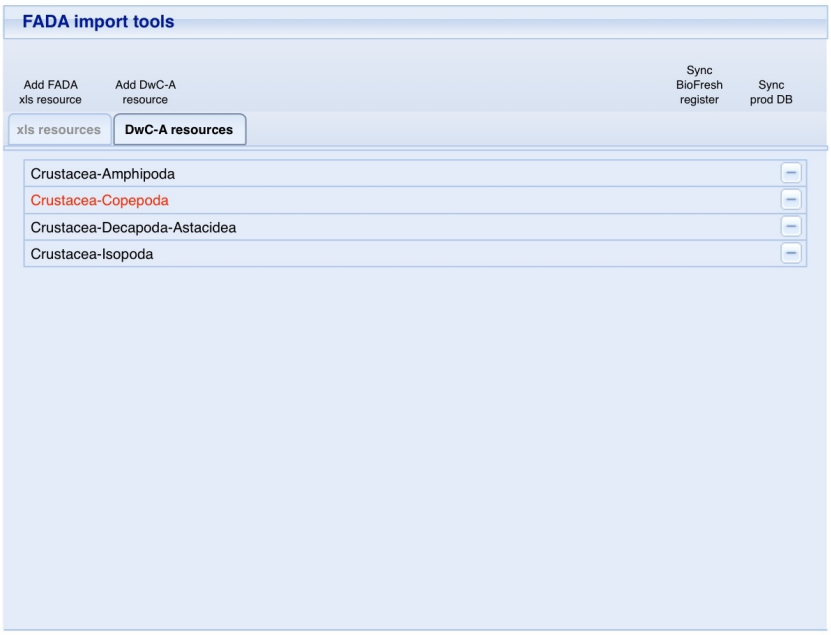

Fig. 4: The mock-up of the general UI showing the organisation of the two modules in tabs.

- The FADA import tool is by definition a backend tool and requires password protection. A single or fixed user/password (database managers only) should be sufficient in the framework of the current project.
- The user interface should distinguish between the two types of resources: Excel imports and DwC imports (different tab).
- The application will allow the management of data sources (Excel files and DwC-A-files).
- The import process will be independently trigger-able for each resource and associated data source .
- For each resource and associated data source a processing log will be kept, reporting at least the number of lines processed, the number of errors, status information on the processing step indicating successful completion or an error message in case of failure.
- The different import tasks have to be performed sequentially, but are actually independent tasks. The operator can stop between any of these step and resume processing at a later stage.
- During the import phase, the user will be presented with a progress bar to indicate which steps have been completed and which tasks are left.

- The previously developed Data Portal Import Tool (DPIT) uses ExtJS version 3.4, the investment of moving to a more recent version (4.2 or up) should be considered in relation to the potential for integration with and re-use of code from the DPIT on one hand vs. the future cost of maintenance.
- General workflow: Similar to the DPIT, the FADA import tool acts on a staging database, which can easily be copied/synced with the production database. If feasible, we would opt to use the same staging database.
- [Clarification] As mentioned in Fig. 8 *"also show groups that are present in the db but have not previously been created/processed through the web tool",* it would be nice if a resource (without associated job) could be created in the database, so it shows up in the interface. As all groups currently present in the database were imported from xls files, this is only relevant for this resource type. On 28/8/2015 it was agreed to implement this option unless it would conflict with other requirements of the app, would require too much work or change the current logic of the app.
- [Clarification] As mentioned in Fig. 4, the "Sync BioFresh register" could be considered as a resource specific function or a global one. During discussion on 28/8/2015, it was agreed that implementing this functionality as part of the resource job was the most logical option.
- [Clarification/New] While this is implicit in several of the mock-ups (Fig. 6 "re-download", Fig. 12 "re-upload file" and Fig. 11: "Re-upload file"), this was not clearly mentioned as a functionality in the specifications. On 28/8/2015, we agreed that there is a need for a job wide abort option, which allows the operator to re-download/re-upload a (corrected) file.
- [Clarification/New] The general UI showing the resources should show all groups for which species/taxon data exists in the database. In order to achieve this, an "initiate script" should be run when first deploying the app, to ensure that the resources associated to these groups are created. To detect the availability of checklist data for a group, this script could consult the field "inputfile_publishable" in fada.groups (WHERE inputfile_publishable='Y').

4. DwC-A processing module specifications

4.1 Envisaged data input format and file location

Input files for Darwin Core-Archive (DwC-A) will obviously be a zipped archive as specified by GBIF (see here), but will be restricted to files with a "taxon core" and not all Darwin Core fields are intended to be imported in the database. We have selected

the required fields for data exchange in the framework AquaRES and have received a sample export from VLIZ. See the ./DwC-file_processing/DwC-AquaRES_field_selection.xlsx file. Selected data fields and extension files will not be mapped to the FADA database, e.g. “nomenclaturalCode” and the “vernacularName” extension. Import scripts have to be constructed. An example export from WoRMS can be provided on request.

The DwC-A files will be posted on a web address. From our side this may be on an IPT (but these are exports from us and thus do not need to be read by the tool), but I doubt this will be the case for VLIZ. As we are still discussing this, we could of course suggest a way which would make our life easier.

4.2 Main processing steps

DwC-A specific processing

- creating resource and validating metadata (see 4.3)
- data load (see 4.4)
- validation (see 4.5)
- inject data (see 4.6)

Processing shared with Excel template processing (see section 6)

- (re)linking to ‘biofresh key’ tables and re-generation of genus-to-families table (but ideally part of the data injection)
- propagation of changes to BioFresh species register
- synchronisation of staging and production database

4.3 Creating resource and validating/editing metadata

Fig. 4 shows the tab for the DwC-import module. Hitting the “Add DwC-A resource should bring up a dialog to choose the concerned organism group (dropdown), this will present a filtered list of the groups for which no data exists in the database yet. See Fig. 5 below. Given that the dropdown menu will present a filtered list of groups for which no data exists in the database, it will not be possible to create an already existing resource and the warning message in the lower left pane can be ignored.

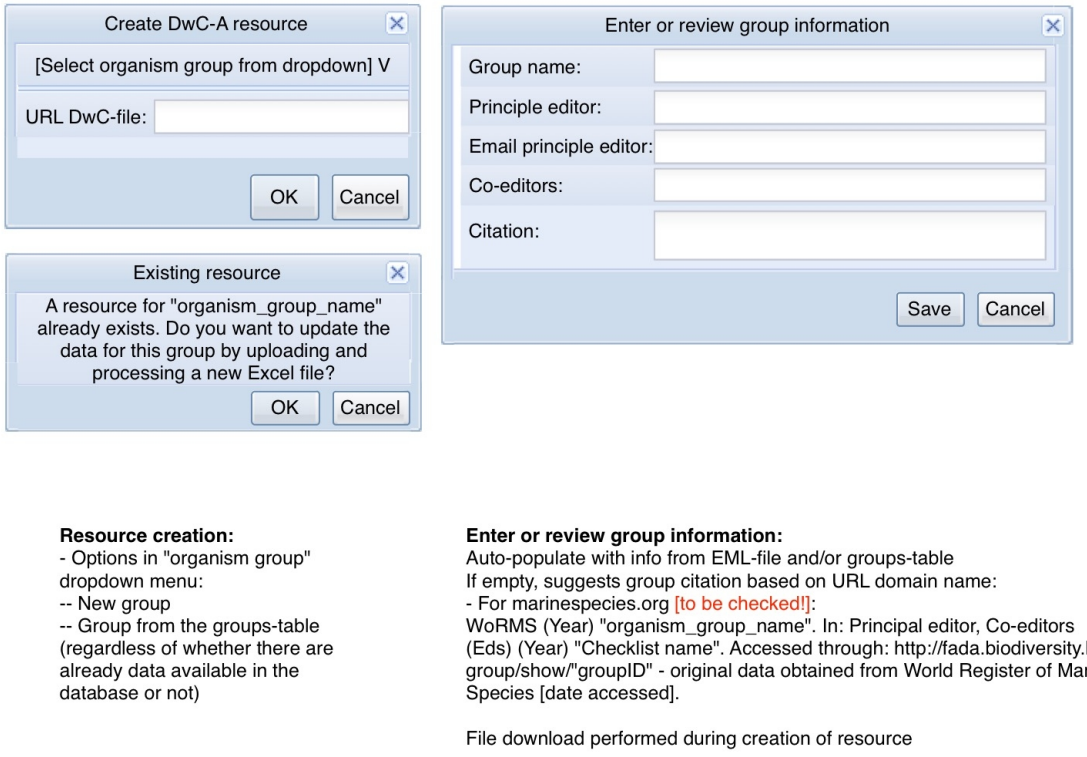

Fig. 5: mock-up of the resource creation dialog. [Warning message in the lower left pane is no longer relevant, see changes in text.]

At this stage, the operator should also be able to manually enter the metadata or update the information available in the database. Potentially, this information could be harvested from the EML ("principal editor" = creator:individualName:givenName + surName; "email principal editor" = creator:address:electronicMailAddress, "co-editors" = concatenation of associatedParty:individualName:givenName + surName; and "citation" = citation), but this would mean older metadata get automatically overwritten, and as we did not foresee the functionality to compare the database content with the EML content, I believe it would be safer to do this manually. ~~validate that the EML metadata contains the necessary metadata for completing details in the FADA group table (check/enter for FADA specific metadata: editor, co-editor, checklist name,...).~~

Once finished, the operator should be presented with an overview of the resource and the pending processing steps as in Fig. 6. In this window, we also envisage an “Edit resource metadata” option (along with “View import logs” and “Delete resource”) which allows the editor to edit the resource metadata at any stage. This could be achieved through a “edit group metadata button” as suggested by Sylvain

Renaudier in his email from 11/01/2016. As discussed under 6.1, this metadata editor would be identical for the two modules.

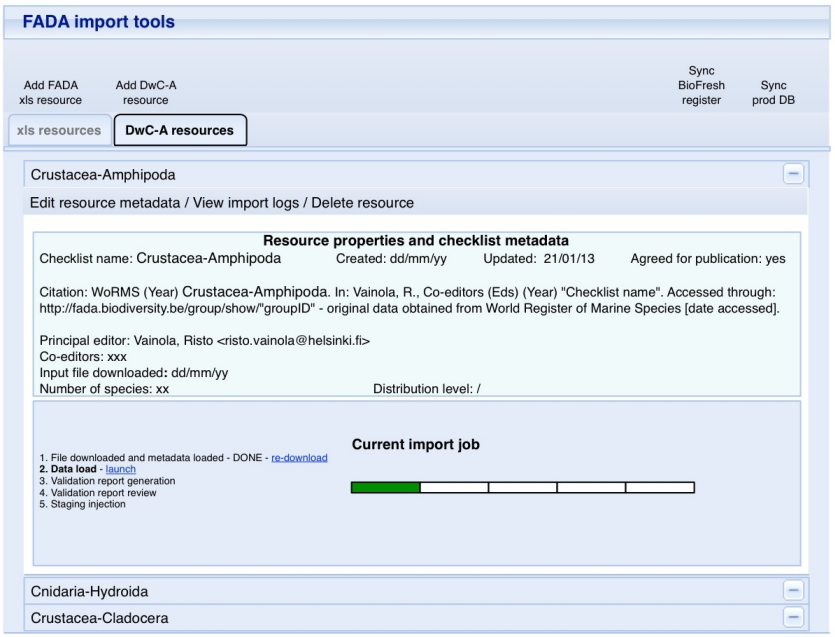

Fig. 6: Mock-up showing the overview of the resource processing status.

4.4 Data load

Load data in flat taxon, speciesProfile, Reference and Distribution table in staging area.

New: Before proceeding to the next steps any species with the isFreshwater flag “N” or the isExtinct flag “Y” should be deleted.

4.5 Validation

Data in staging tables

Note: while the validation will act on different tables and fields, and might be slightly different for the DwC-A vs. xls processing, the “business logic” and validation steps are the same, and can thus be organised in a way that the validation steps can be called by the two processing modules.

- The first “checks” are not really part of the validation process, but are rather “data cleaning’ to facilitate further processing. These modifications can be performed automatically and reported in the logs, but do not need to generate a warning or require intervention of the operator. These include;
    - Eliminate empty rows
    - Eliminate duplicate rows (keeping only first one
    - Eliminate characters ‘\n’ ‘\t’ ‘\u00A0’ and multiple blanks
    - Check for accidental use of “o” instead of “0” and vice versa. E.g. use of “o” in year (199o), use of “0” in latin-only fields (spin0sus). Replace where detected.
- Check whether mandatory fields are present (see overview of DwC-field recommendations as discussed under 4.1) - if error WARNING **mandatory field(s) missing** field_name, no further processing possible

  Note: Originally the Excel overview was constructed more from the perspective of the data provider. During a discussion with Sylvain Renaudier on 6/8/15 we realised that it would not be useful to generate warnings for every single missing field which is labeled as mandatory. This designation as “Mandatory” was therefore reviewed and updated in the Excel-file ./DwC-file_processing/DwC-AquaRES_field_selection-with_FADA_mapping-v1.1.xlsx.

- Check whether content of fields corresponds to the expected format - if error WARNING **format error** and show line + highlight field, present dropdown menu allowing to ignore/consider empty/edit the field;
    - namePublishedInYear should be of the format 1999a, i.e. 4 digits and optionally one latin character [a-z]. The 4 digits should be >1730 and <current year + 1.
    - The fields kingdom, family, genus, specificEpithet, infraspecificEpithet should only contain latin characters [a-z], accented characters are not permitted
    - The field subgenus should be of the form “Genus (Subgenus)”
    - The field taxonRank should correspond to the control vocabulary: “subspecies”, “varietas”, “forma”, “species”, “genus”
    - The field taxonomicStatus should correspond to the control vocabulary: “invalid”, “misapplied”, “homotypic synonym”, “accepted”
    - Check whether the field datasetID contains an URL
    - The fields isMarine, isFreshwater, isTerrestrial, isBrackish and isExtinct, isPreferredName should contain boolean values or values that can be translated to booleans 0/1, t/f, TRUE/FALSE, y/n

    - language should correspond to the ISO 639–1 language code and countryCode should be 2–3 characters (ISO3166 alpha 2 (3 is permissable) country codes)
    - startDayOfYear and endDayOfYear should be between 1 and 365
- Checking for unique taxonIDs in taxon table - if error WARNING **non unique IDs**, show lines/list IDs, offer possibility to ignore a line or abort the import process
- Check whether acceptedNameUsageID and parentNameUsageID refer to taxonIDs present in taxon table - if error WARNING **core IDs missing from taxon table**, show lines/list IDs, offer possibility to ignore a line or abort the import process

  Note: The GBIF library for processing DwC-A automatically ignores the entries in the extension files that do not match the core IDs. As this procedure also ensures the relational integrity and it would require (potentially time consuming) code forking to perform this check, we decided to skip it on 6/8/15.

- Checking whether taxonIDs/coreIDs in extension tables correspond to taxonIDs in taxon table - if error: as above
- Checking the consistency of the row, e.g.: if the “specificEpithet” field is provided, the “genus” field cannot be empty. If error WARNING **line consistency problem for** ‘taxonRank’ missing ‘higherTaxonRank’: show line and offer possibility to ignore line or edit line. Below is a list of fields that have to be filled for a given taxon rank:
    - infraspecificEpithet (taxonRank= “subspecies”, “varietas”, “forma”): specificEpithet, genus and family have to be (minimally) provided
    - specificEpithet (taxonRank= “species”): genus and family have to be (minimally) provided
    - genus (taxonRank= “genus”): family has to be (minimally) provided
- Check whether data for higher taxonomic levels are correctly declared; genus requires family, species requires genus, subspecies requires specificEpithet. - if error WARNING **higher taxonomy missing in input file for** taxonRank:scientificName missing taxonRank:scientificName, offer possibility to ignore or edit line. See Figs. 7 for an example mock-up. The levels that have to be declared are the same as for previous check, although here it concerns the valid declaration of a specific taxonRank at line/record level. For DwC-A files this extends to the higher taxonRanks as follows (although missing content for these levels is not considered critical);

- family: order, class, phylum are expected
- order: class, phylum are expected
- class: phylum is expected

Additionally, for data provided using the Excel-template, if applicable, the intermediary taxonRanks (optional ones in square brackets) have to be checked, e.g.;

- Subspecies: Species, [Species group], [Subgenus], Genus, [Subtribe], [Tribe], [Subfamily], Family
- Species: [Species group], [Subgenus], Genus, [Subtribe], [Tribe], [Subfamily], Family
- Species group: [Subgenus], Genus, [Subtribe], [Tribe], [Subfamily], Family
- Subgenus: Genus, [Subtribe], [Tribe], [Subfamily], Family
- …

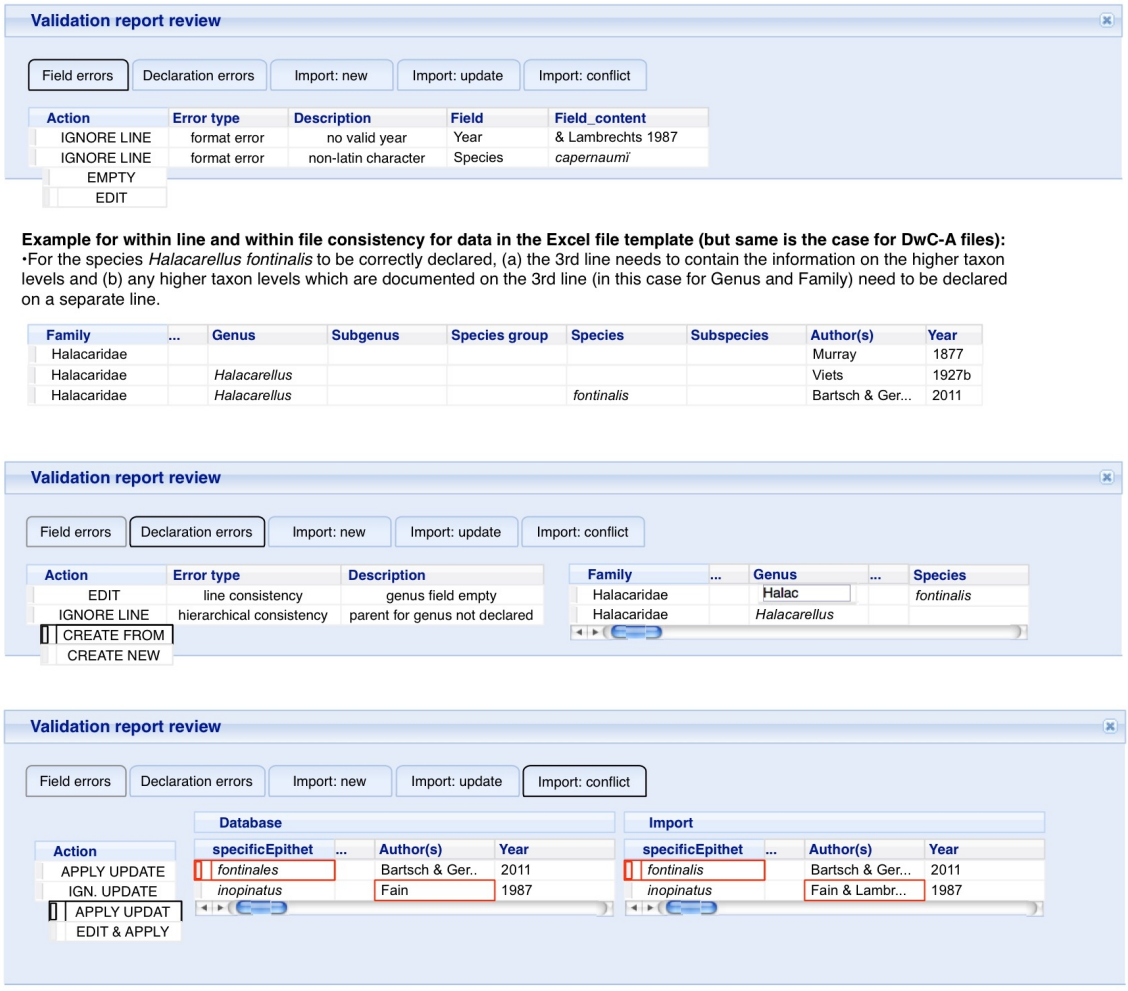

Fig. 7: Mock-up of the validation interface

During discussion with Sylvain on 6/8/15 we realised that the combination of record (ignore line) and field (empty, edit) level actions as envisaged in the mock-ups (Fig. 7) is not ideal and could result in potential conflicts (e.g. ignore line + edit field action for same line). As a solution, we agreed to split this up as follows; 1) report the lines with errors and the type(s) of errors per line, and offer the possibility to ignore the lines (=delete the content from the import) or act on the line = “process line” (the choice between processing individual errors on the line “as is” or act on the individual fields affected is part of step 2) and 2) provide the option to act on specific fields containing errors, providing the options ignore error, empty field and edit field.
Following this logic, the processing of the declaration errors and import conflicts also needs to be re-considered. The “line consistency errors” could probably be reported along with the other record level error reporting (as described in preceding sentences), while the hierarchical consistency errors require a separate interface to add the missing info (cfr. the “create from” and “create new” option in the middle pane of Fig. 7). The form for entering this info does not need to cover all DarwinCore fields, but requires most of the info from the taxon sheet to be entered/generated; *Family, Subfamily, Tribe, Subtribe, Genus, Subgenus, Species group, Species, Subspecies* (depending on the taxonrank for which there is a hierarchical consistency error; i.e. if error is for undeclared Genus, only the rank Genus and up need to be completed), *Author(s), Date, Original genus, Original species name, Parentheses* and *ref key* have to be included in the form.
It was agreed that the implementation of the “validation report review” which is presented in 5 tab sheets in Fig. 7 will be split up in 3 stages; 1) Hierarchical consistency errors (only covering the last point under 4.5 Validation), 2) Record and field level errors (itself split up in 2 steps as described earlier) and 3) import status errors or “import preview”, each with its specific interface and options for dealing with the warnings/errors.
In summary, the following stages will have the processing options;

1) Hierarchical consistency errors: ignore line, create [parent record] from [info in child record], create new [parent record]
2) A. Record level evaluation: “Ignore/Delete” OR “Process” (the latter being the default option) B. Field level evaluation: “Ignore error” (= process “as is”), “empty” (this could also be achieved through editing of course, so if this would programmatically be easier, this option can be dropped) OR “edit”

Note: Once the report is validated, these changes will be directly be propagated at DB level on candidate DwC-A/Excel import records (delete, update…).
@Sylvain: Please let me know in case you were expecting more details “in terms of UI interaction” and “scenarios”.

Imported data compared to data in the database tables

Note: this section (4) is specific for the DwC-A processing, in which case we can rely on the availability of the (provider)taxonID or coreID. For Excel-dataprocessing a similar logic needs to be implemented based on the elements of the scientificName.

Note: In some cases the “provider” part in “providerTaxonID” may be missing, the reason for this is that this refers to the term as used in the DwC-A-file as we receive it. To avoid confusion with the taxonID used within the FADA database, it is safer to refer to the providerTaxonID when referring to the taxonID as mentioned in the DwC-A-file.

[Rewrite attempt according to “algorithm logic”]

The algorithm could be organised as follows:

1) Identify whether the imported data concerns an entirely new group (based on the groupID). If so, all data is new and there is no need for comparison to data in the database.

2) Else, check whether the providerTaxonID is new or is associated with a record that has previously been imported in the database. As discussed in the documentation on the database changes under point 2.5, this would require a table the import schema to keep track of the provider taxonIDs from previous imports. If new AND taxonRank is species or subspecies: check whether any of the following subsequent checks yield a match:

- Search “scientificName” in the species or synonyms table
- Check whether the combination/concatenated names of the “genus” “specificEpithet” and “infraspecificEpithet” can be found in the species or synonyms table
- If the taxon/scientificName is not detected using exact matching during the 2 previous steps, repeat with phonetic matching.
- If no matches are found, the record is considered NEW and should end up in the corresponding list, which allows the operator to see what’s new.

If a match is found during any of these checks, the corresponding records should be displayed among the **POTENTIAL CONFLICT** records

3) Else, if **providerTaxonID is existing AND taxonRank is species or subspecies**: Validate whether imported data compared to the data in the database' species and synonyms table have the same "scientificName, acceptedNameUsageID, parentNameUsageID, acceptedNameUsage, originalNameUsage, parentNameUsage, namePublishedIn, namePublishedInYear, kingdom, phylum, class, order, family, genus, subgenus, specificEpithet, infraspecificEpithet, taxonRank, scientificNameAuthorship"

If so, these records can be considered be **UPDATED or UNCHANGED**

Note - for discussion: Originally I envisaged that the app would detect what has changed, so you can see/validate as an operator what has changed, but maybe this would overcomplicate things?

4) Else, if **taxonRank is species or subspecies AND any of these fields has different input compared to the data in the database**, these records should be displayed among the **POTENTIAL CONFLICT** records. The interface should allow the operator to visualise for which field there is a conflict and have the options to ignore (keep former data), override (adopt new data) or edit and override (adopting newly entered data).

5) If (3) does not apply, so in the case: **providerTaxonID is existing AND taxonRank is NOT species or subspecies**: Validate whether imported data compared to the data in the database' taxon table have the same (scientific)name.

If yes, these records can be considered be **UPDATED or UNCHANGED** , else these records would be considered **POTENTIAL CONFLICT** records.

Note - for discussion: Originally I also discussed the entire higher taxon levels problematics. To be honest, I am not entirely sure how to deal with those, so the easiest solution is probably to deal with these issues ad hoc directly acting on the database.

~~Based on the groupID (entirely new group?) and (provider) taxonID/coreID > Check which data are already present in the database and compare content of fields if the provider taxonID/coreID is already present. Note that providerTaxonID is a new field that needs to be added to the database. As mentioned above this comparison is now part of a separate validation step, the "import preview".~~

~~Records can either be;~~

NEW: Alert the operator that this is new (unless it is an entirely new group - a group is considered new if no species data is associated to it. At the level of the import app, this would mean that there is no resource, either DwC-A or Excel import associated to it).

UPDATED: Associated information added (e.g. distribution and speciesProfile data previously not available). Alert operator, default option "Apply update", option for operator to "ignore update".

POTENTIAL CONFLICT:

This type of error mainly applies for any name related fields including taxonID, scientificName, genus, specificEpithet, etc. covered by the next checks.

In case of errors, the operator should be presented with the options to ignore (keep former data), override (adopt new data) or edit and override (adopting newly entered data).

- Check whether records with the same provider taxonID (and resourceID) in the species and synonyms table have the same "scientificName, acceptedNameUsageID, parentNameUsageID, acceptedNameUsage, originalNameUsage, parentNameUsage, namePublishedIn, namePublishedInYear, kingdom, phylum, class, order, family, genus, subgenus, specificEpithet, infraspecificEpithet, taxonRank, scientificNameAuthorship"
  Note: the resourceID is linked to the import application tables and is expected to be referenced in the groups table as a foreign key. The groups table has to be updated accordingly.
- Note: The file ./database-info/Field_mapping.xlsx provides an overview table with the field mapping.
- If the taxonID is new, check whether the "scientificName" can be found in the species or synonyms table and whether the status is "accepted" for names in the species found in the species table and "invalid" for names found in the synonyms table.
- If the scientificName is not detected, check whether the combination/concatenated names of the "genus" "specificEpithet" and "infraspecificEpithet" can be found in the species or synonyms table and whether the status is "accepted" for names in the species found in the species table and "invalid" for names found in the synonyms table.

- ~~If the taxon/scientificName is not detected using exact matching during the 2 previous steps, repeat with phonetic matching.~~
- ~~Check whether scientificNames with taxonRank =< family are new or are updated (for a new resource this should be the case, but during an update, the operator would be interested to know what's new and what was already present in the database).~~
- ~~Check whether scientificNames with taxonRank > family need to be added or are already present in the taxon table (check for conflicts) - if error WARNING **conflicting higher taxonomy**, show conflicting levels + names, provide options: ignore (keep original higher taxonomy), override (adopt new higher taxonomy) and edit and override (edit and adopt new higher taxonomy).Currently the higher taxonomy is copied from MySQL database tables on the annual release of the Catalogue of Life DVD. A more sustainable solution for this issue, using the Catalogue of Life web services should be investigated.~~

See lower panel of Fig. 7 for a mock-up example of the validation interface for conflicting changes.

Note by MK. On 2016-06-20 Sylvain and I discussed the difficulty of finding back ancestors in DwcA files. We concluded that the information provided in DwcA files makes it more coherent to use the parentNameUsageId to find the taxonomic parent organism.

4.6 Data injection

Inject data in database tables and update biofresh_key tables. Updating the biofresh_key table requires to build an overview of the updated names (e.g spelling corrections), which need to keep their original biofresh_key, the new species, for which a new id is generated, and the deleted ones, for which the logical delete flag needs to be set. See more background details on the biofresh_key tables under 2.4.

The tables need to be filled in the order taxons > species > synonyms, to allow the latter tables to reference the former. The taxon table itself also needs to be filled in a hierarchical order, so lower taxonomic levels can reference the higher ones. More specifically, families are added first, followed by all declared sub elements down to the genus. Currently, the *Original genus* and *Declension species* are added to the taxon table before adding the (accepted) *genus* (and any lower levels), but as these entries are not (necessarily) linked to a parent, this order is probably less important.

While adding the species and subspecies taxons table, it is possible to create the species and subspecies records in the species table in parallel.

Note: the scientific_name is “calculated”/concatenated as follows: Genus+” “ if provided “(“+Subgenus+”)”+ “ “+Species+” “if applicable+Subspecies+” “if Original genus is provided (parenthesis are “Y”)”(“+Author(s)+” “+Date+if Original genus is provided (parenthesis are “Y”)”)”. If present, the species group should also be added as “[species group]” in between subgenus and species. Similarly, the “genus_species_name” is the cannonical species name (without authorship info!), this is a concatenation of the following fields: Original genus [Synonym]+” “+ Species/Subspecies [Synonym] OR if synonym for a genus Genus/Subgenus [Synonym] and is set during injection.

Once all taxon elements and species are created, we currently launch the process to link groups to a specific taxon level. This is done by identifying the highest taxon rank which is unique to the group. This information is important for the FADA app, to identify the highest taxon level to be shown when browsing the group.

Finally, the synonyms are processed and stored in the synonyms table.

5. Excel template processing

While scripts for importing data are currently available for importing Excel data, it would probably be more efficient to re-write them. Nevertheless, these scripts could provide inspiration for how to tackle specific issues. More details can be found in the “Current workflow: Excel template processing” section of an earlier version of the specifications document FADA-import-specs.md.

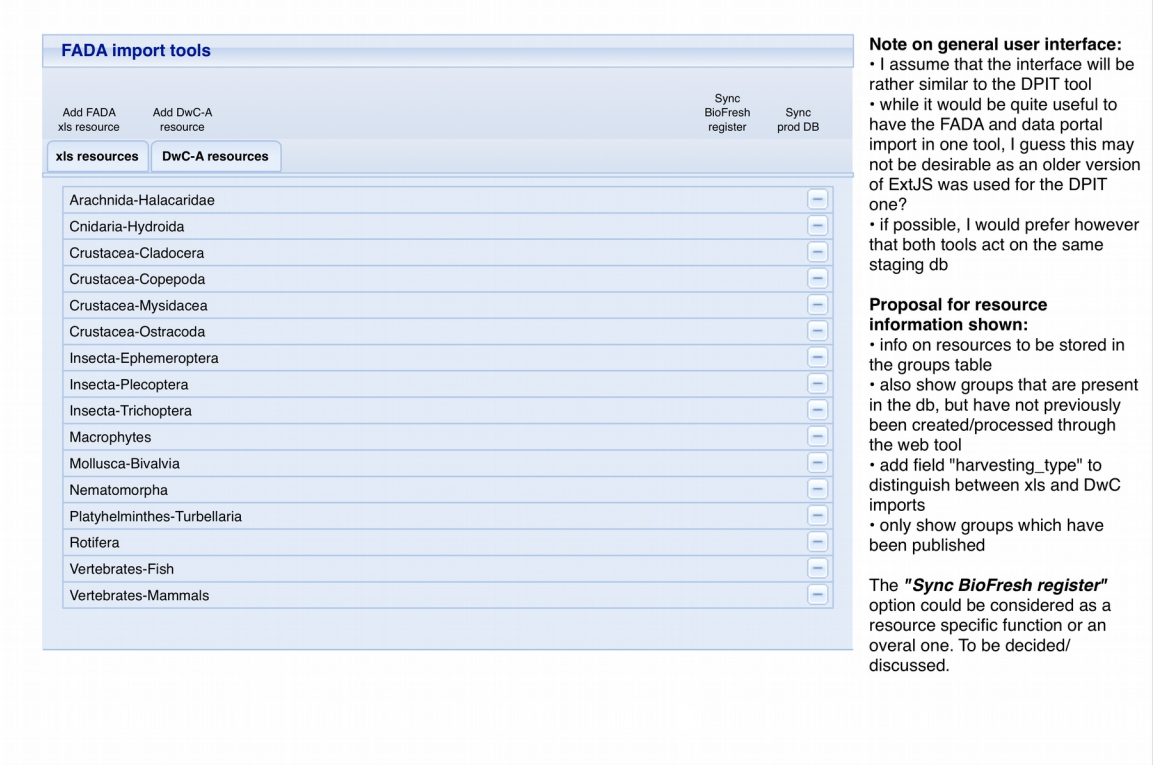

Fig. 8: Mock-up of general UI for Excel resource processing.

5.1 Envisaged Excel input format

Excel-data are provided in an excel-template, for which there are currently 2 “recognised” versions. A slightly modified version (**2.1**) including the columns isFreshwater, isMarine, isTerrestrial and isBrackish will need to be prepared reflecting the planned database changes (see 2.3). While editors previously had the possibility to provide checklist data through multiple files for a single group/resource, we may choose to abort this option for simplicities sake (if necessary we can still manually join the files).

Note: as specified in the FADA-database-changes.md document, specific fields (*exotic* and *bodyMass*) and worksheets (conservation) are not stored in the database.

5.2 Main processing steps

xls specific processing

- creating resource and validating metadata (see 5.3)

- checking column mapping (see 5.4)
- data load (see 5.5)
- validation (see 5.6)
- inject data (see 5.7)

Processing shared with DwC-A processing (see section 6)

- (re)linking to ‘biofresh key’ tables and re-generation of genus-to-families table (but ideally part of the data injection)
- propagation of changes to BioFresh species register
- synchronisation of staging and production database

5.3 Creating resource and entering metadata

See the UI-mock-ups in Fig. 9 and 10.

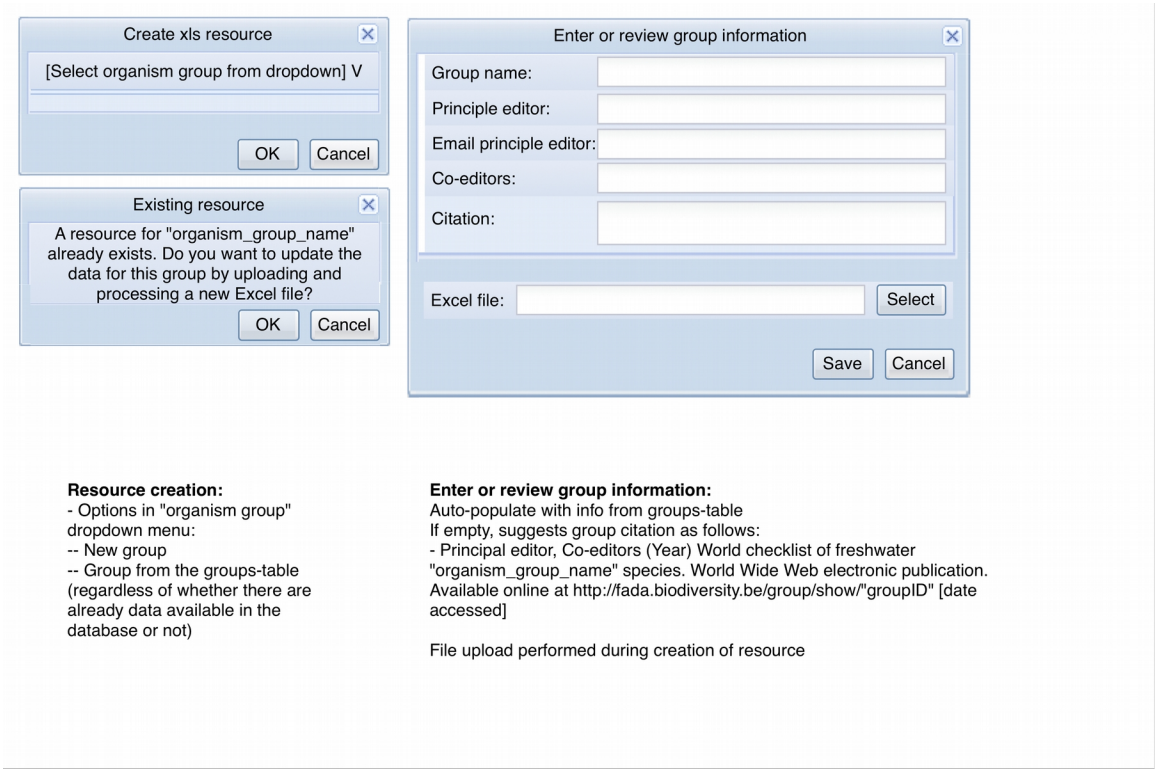

Fig. 9: Mock-up of the Excel resource creation dialog.

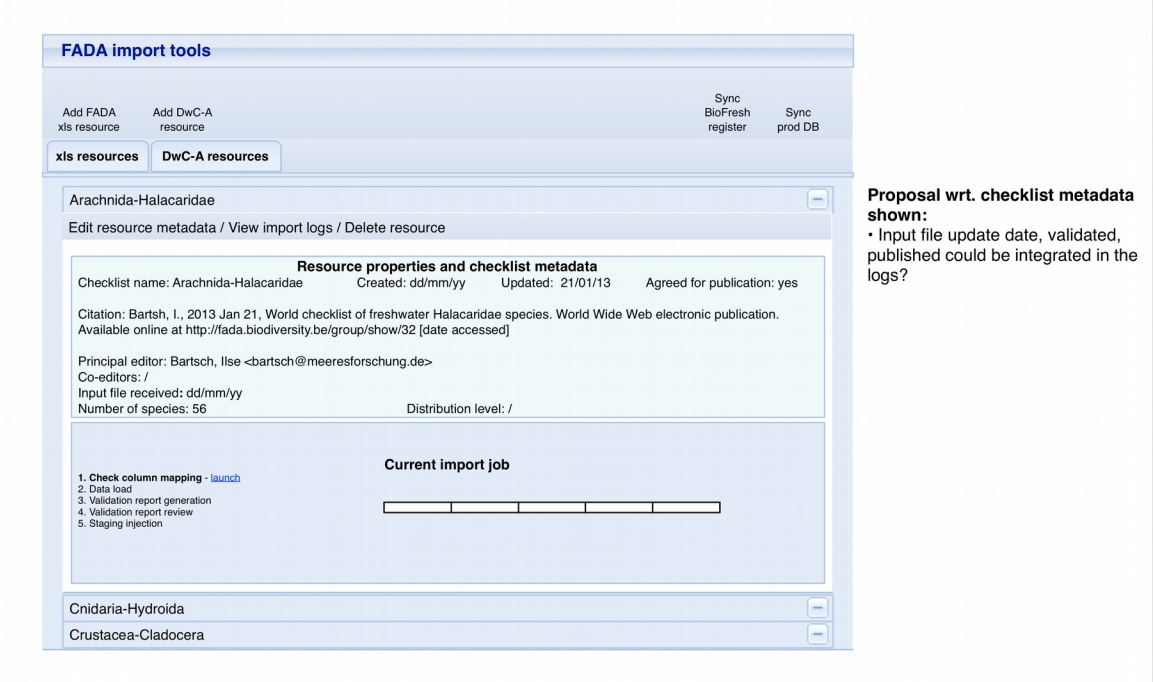

Fig. 10: Mock-up of resource status overview after creation.

During the creation of an Excel resource, the operator can create a resource for an organism group present in the groups table (cfr. http://fada.biodiversity.be/group/list?current_page=groups) or create a new (sub)group as detailed for DwC-A resources under 4.3.

In case of an existing group for which information is currently available, the operator will be presented with a warning to highlight the fact that this resource already exists. Further processing will not create a new resource, but continue with the review of metadata and upload of an Excel file using the existing info on the resource (also allowing it to be updated).

For a group present in the list, metadata present in the group-table will be included in the dialog and can be reviewed and completed at this stage. As the Excel-file does not contain checklist metadata, these fields will be empty and need to be filled for a new resource. With the exception of the field “co-editors”, this information is mandatory.

Note: The information on the Principal editor is stored in the *users*-table rather than in the *groups*-table. From the perspective of the import app, this means that, unless the name entered in this field exactly matches an entry in the *users*-table, a new user will be created.

We propose to include file upload (from the operator's computer) as part of resource creation. In case of file upload errors, the application should offer a retry option before closing the resource creation window and process.

Note: From a practical perspective we will split this in two stages again: 1) resource creation (only selecting the organism group from a dropdown) as a separate step from 2) data download and metadata creation/upload. This roughly corresponds to Fig. 9, where the upper left dialog would represent a separate processing step.

5.4 Checking column mapping

The use of Excel files makes it impossible to be certain of the structure of the files that are sent to us. Contributors can make errors in data structure, field positions, start of data in excel sheets, etc. It is therefore necessary for the operator to make sure that the files comply with one of the two templates that have been agreed on. Essentially this is a check of columns and data position. The easiest solution seems a quick visual checking mechanism to validate the field mapping. This is illustrated in Fig. 11 and 12.

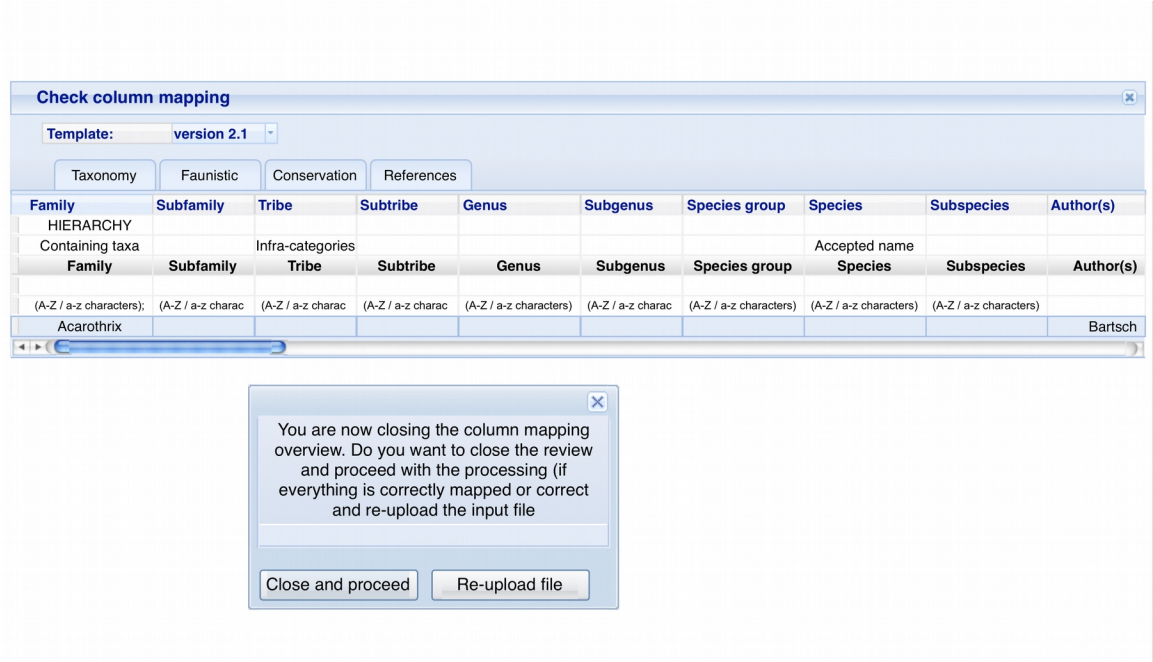

Fig. 11: Mock-up column mapping interface.

Note: Fig. 11 shows a multi row comparison for the visual column mapping tool. For practical reasons it was agreed (30/7/15) with Sylvain that only looking at row 3,

which contains the column headers, would be sufficient. The only issue with that is that the label for RefKey is on row 2 and row 3 for this column is empty. This will be corrected in the new 2.1 template version and earlier versions of the template that are sent out before finalising this version. Nevertheless, during the check this header field is potentially empty (and should not be considered as an error).

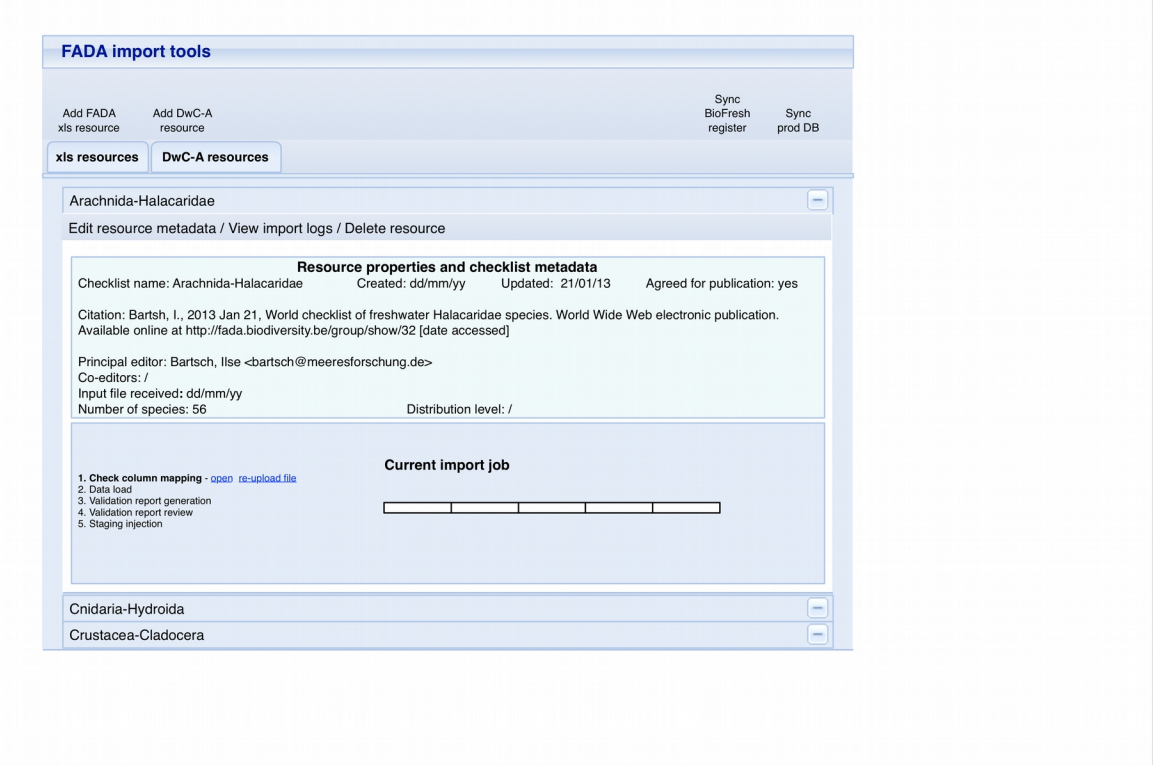

Fig. 12: Mock-up of the resource status overview after completion of the column mapping.

5.5 Data upload

The upload step is the reading data from Excel files and storing it in tables in a staging area.

Upload happens per group. The files to process are all located in a directory specific for the group/resource (named after the group's name) and all loaded one by one.

The upload process looks for three sheets of data are identified based on their names, which can already be validated while checking the column mapping. Each

sheet – “Taxonomy”, “Faunistic” and “References” respectively – is stored in a dedicated import table (taxa_table, distr_table, ref_table).

Note: Currently the taxon rank for each record is calculated at upload, which aids further processing.

5.6 Validation

Data in staging tables

- For data entered in the Excel template, we do not use the notion of “mandatory fields”. Nevertheless, it would be useful to flag whether author and year are empty on specific lines. This could be flagged as **recommended data missing**: ‘year’/‘author’ not provided for line xx.
- While the data are rather different in terms of organisation, validation for **format errors** the corresponding fields is the same as 4.5.
- If present in the “References” the presence of the Refkey in the “Taxonomy” sheet has to be checked.
- As data in Excel format does not come with “Core IDs”, the appearance of the names from the “Faunistic” sheet has to be checked.
- As in 4.5, there’s a need to check the **line consistency** and whether data for **higher taxonomy is missing**.
    - Particular for the Excel templates is that synonyms are not declared in the same format as the accepted species, but are present in dedicated columns. To be validly declared, the accepted species to which the synonym refers has to be declared on a separate line (similar to higher taxonomy levels). However, as this is a easily overlooked by the editors, I propose to implement the option to **declare species based on information on line xx** as default option.
    - Another line consistency check particular for data in the Excel-format is checking whether the “original_genus” is provided when the parentheses flag is set to “yes” and/or if “declension_species” is given. As we now also aim to store the parenthesis flag, this check should also be done in the other direction, i.e. verify whether parentheses flag is set to “yes” if a “declension_species” or “original_genus” (different from the “genus”) is provided.

Imported data compared to data in the database tables

The validation in comparison to data already in the database is also similar to 4.5, with the exception that it cannot be done based on IDs, and can exclusively be performed through name matching.

Currently the “import preview” is primarily focusing on the taxons table. The first step is to construct an in-memory version of the taxons table (covering the name, rank, group_id, and parent_id fields) based on the imported data (“as these data would be injected”). This is done by constructing a tree for all families within a group starting at the family level and drilling down to species/subspecies level as applicable.

Once this in-memory table is built, it can be compared to the data present in the fada.taxons table to detect which entries are new. Currently this comparison itself is also organised in a hierarchical manner, i.e. starting by checking the presence of families, subfamilies, and moving down.

This comparison is also performed in reverse, i.e. checking which entries from fada.taxons are not present in the in-memory table, to detect entries that have been deleted in the updated checklist. We propose to implement a similar approach, but additionally check whether the author(s) and year are identical or updated.

A full comparison of (species present in) the imported taxon tables with the fada.species table is not absolutely necessary as the content overlaps strongly with that of the taxon table. In the current processing, such a check is only performed for the original species names (Original genus + declension species/species). Similarly, the checking of the entries in the synonyms table could be restricted to the information in the genus_species name (and possibly author and year) to check whether these names are present.

Additionally, a similar process as for the taxons table should be implemented for the faunistic information (including the newly added environment flags). This could be done by constructing an in-memory version of the faunistic table for comparing with the information in the fada database.

Identifying whether a record in the species table concerns the same species as in the imported data should be done based on name matching (which is currently done for re-linking the species table and the biofresh_key_species table) and is based on a concatenation of Genus, (subgenus), species, subspecies/infraspecificEpithet. Author

and year are currently not considered in this check, but should be compared in a next step to identify any updates to the information in these fields.

If a name is not found during these comparisons, a phonetic match/matching based on Levenshtein's distance, offering the possibility to "use to update suggested taxon" (cfr. the link to suggestion option in the DPIT-tool) to the operator would be ideal (as mentioned under 2.4 and 4.5), so the original IDs for these entries can be maintained.

5.7 Import

As 4.6.

6. Joint components

6.1 Resource metadata entry and edit module

Similar to metadata inspector/editor under 4.3 but including all fields from the dataset table (e.g. abstract/description) and the new ones described under 2.3.

6.2 Synchronise data with BioFresh species register

Note: this sync is preceded by the synchronisation between the fada.taxons/species/synonyms table with the biofresh_key_taxons/species/synonyms tables. As described under 2.3 the details of this sync process depend on the choices made with regards to database refactoring, and as mentioned in 4.6 ideally are catered for during data injection.

New: As the FADA database potentially contains non-freshwater taxa (only for data imported in Excel format) and these are not supposed to be propagated to the BioFresh species register, these entries have to be filtered out / disregarded when syncing the data.

As a final step in the resource job, there is a need for synchronising the data imported into the FADA database/schema with the BioFresh species register table. In addition to the requirements needed for updating the biofresh_key tables, this also requires an overview of the updated “original combinations”*. In contrast to the species and synonyms, which are stored in a specific table, there is no separate table for these “original combinations”. A list has to be constructed by combining the information from the following fields; original_genus, declension_species or species (if the former is empty), year and author.

As mentioned under 2.4, for new species names added to the FADA database, their presence in the BioFresh species register needs to be checked (as this register can be populated through other sources) based on the group and species name matching (both exact and phonetic).

This will likely require an interface for validating the changes; (a) accept/ignore/link to existing for new entries, (b) check exact matches and accept to update “name source” to FADA, and (c) check phonetic matches and update name from FADA/add FADA name and keep name from register as synonym.

FADA data compared to data in the BioFresh register

Checking which entries are new, deleted and updated is similar to the process for the “preview” for importing data into the FADA database (The “Imported data compared to data in the database tables” section under 4.5 and 5.6), with the exception that it should additionally consider the “original combinations” as mentioned above. Matching between the entries in FADA and the register (register.biofreshspeciesregistry) can be done based on the biofresh_key and register_id (link established in the register_to_fada table).

For “original combinations”, their presence in the register can be checked using two methods;

1) by looking for entries where the register_id of the corresponding (accepted) species is stored in the relatedname_id field and the taxonomicstatus_id is set to 4 (which can only the case for 1 entry) or

2) based on name matching (using original_genus + species/declension_species on the side of FADA and genus + specificEpithet on the side of the register).

Original combinations are not (and cannot be) directly linked to FADA through the register_to_fada table, but instead resolve through the accepted species by storing the corresponding register_id in the relatedname_id field and get a taxonomicstatus_id=4. So, if an original combination cannot be found in the register, a new entry is created and and linked to the accepted name in the register.

* Original combinations are “previously accepted species name” which have become “objective synonyms” because the species has been moved to another Genus. In the Halacaridae example file, the species “Halacarellus hyrcanus (Viets 1928)” has as original combination “Caspihalacarus hyrcanus Viets 1928”. For some reason, these names are not stored in the synonym table, but are only referenced through a taxonID in the species table.

6.3 Synchronise the staging database with the production database

In parallel to the developments of the FADA import tools, we will work out a solution to improve the synchronisation/replication of the staging database (for both the DPIT and FADA import tool) with the production database. If successful, we will provide the code to trigger the synchronisation for use with the “sync prod DB” button in the general interface.

Note: this is not strictly part of the specifications. The production database synchronisation is indeed handled manually, nevertheless we are still looking (internally) for a better staging - sync setup to handle the database at Belspo and Gulledelle, and were wondering if the PostgreSQL replication functionality would be useful…

The ideal situation would be to set up a staging environment at the BBPF servers at BelSPO, however, we will first explore to set up such an environment at the BEDIC servers at Gulledelle.

The advantage of the former option is that once the staging DB is ready for production, the BBPF system administrator only has to

- delete the prod db

- change the name of the staging db to prod db

- prepare a new staging db by copying of the prod db into a new staging db. Copying first and using this db for production is also possible, but as this requires more time, a simple renaming is easier for the staging -> production step, while the time it takes to copy the database should be less of an issue when rebuilding the staging database.

1. The “operational unit” can be referred to as “Group”, “Data source” or “Resource”. In this document the term “group” is used to stress the link with taxonomic unit, whereas “resource” is used when discussing the digital representation of such a group (e.g. as an ‘import unit’ and its associated file). ↩