Importing data into Tethys

TABLE OF CONTENTS

1 GENERAL CONCEPTS

Tethys supports many ways to add data to a Tethys database. This can be done via an import map (sometimes called a source map) which provides a mapping from a data source (e.g., an Excel workbook) to Tethys fields, programmatically through the data Nilus library which generates Tethys-compliant XML, or via some third-party software tools such as PAMGuard which will support direct export to Tethys in 2024.

In this manual, we describe how to:

- import XML documents directly
- import from databases, spreadsheets, etc., using an import map
- create import maps with a web-based drag-and-drop tool or by composing an XML document
- write code to generate XML documents

Which one should I use? If your data are already using the Tethys standard XML, or if someone has specified an import map for you, you can skip directly to the importing data section. If not, you should ask yourself whether or not your data are well structured. Import maps can accommodate some irregularities in your data. For instance, one can specify a default value for when data are missing or conditionally include sections of a Tethys document based on simple conditions. An example of this might be to conditionally produce information related to duty-cycled recordings only when the instrument had a periodic recording schedule.

2 IMPORTING DATA

If you have a data provider that generates Tethys-compliant XML files (e.g., PAMGuard will do this starting in 2024), you can just import your XML files directly into Tethys.

When you do not have such a provider, you must first follow one of two routes to get your data into Tethys. The first method is to establish an import map, which can be done with either a graphical or text-based interface specification file (sections 3.1 and 3.2 respectively).

Data can be imported using a variety of methods. The most user-friendly method is to use the web-based import, followed by the dbSubmit interface that is available from MATLAB. Finally, a more text-based import is available in the Python client.

2.1 WEB-BASED IMPORT

Tethys web servers support an import data option that can be accessed from the main menu at the top right: “Import data.” The data import screen (Figure 1) contains three sections.

2.1.1 Specifying how data are to be translated

The first section of the page provides details on how data are to be translated into Tethys and contains three items:

- Import map (also called a source map). When data are already in a Tethys-compliant XML format, this dropdown menu may be left alone. When data are to be converted from a spreadsheet, database, etc., you will need to select from an existing import map that details how to convert your data format into Tethys-compliant data. A list of existing import maps will appear in the dropdown menu. Information on creating new import maps can be found in section Import Maps3 on page 15. When an import map is selected, it will automatically select the correct collection to which documents are to be added. It may also add a data source if the import map author has suggested for what type of documents the source map was designed.
- Collection – Tethys organizes documents into themed collections of documents. Collections are described in the general Tethys manual, but commonly added collections are:
    - Deployments – A description of an audio recorder deployment.
    - Detections – A specification of effort to detect sounds and what was found.
    - Localizations – A specification of effort to determine directionality, position, or track of sound producing sources (e.g., animals, ships).
    - Calibrations – Description of acoustic calibration efforts for deployments.

    With the exception of Tethys compliant data, selecting a source map will automatically set the collection. While not as commonly used, new import/source maps, sets of species abbreviations, and ensembles may also be added. See the Tethys manual for further details about these collections.

- Species Coding – This need only be set when submitting data that describes species that can be associated with the Detections and Localizations collections. It may only be used when an import map has been specified. Tethys stores species descriptions internally as integrated taxonomic information system taxonomic serial numbers (TSNs). Any time translation is specified, numerical species data is treated as a TSN. Species names may be submitted using Latin names that match the ITIS records, or users may use/develop their own coding map that associates species abbreviations with Latin names. Details on species abbreviations maps may be found in the main Tethys manual. The dropdown menu will list the species abbreviations maps that are currently available on the Tethys server.

2.1.2 Specifying the data to import

The second section of the import page allows users to specify where their data are coming from. If selecting an import map did not create an entry automatically, the + Data Source button will create an entry. The types of data sources that are available are dependent on the open database connectivity (ODBC) drivers that are available on the Tethys server machine. Speak to your Tethys database administrator if the one that you are interested in is not available, this can usually be resolved by installing the appropriate ODBC driver.

Each data source may have a source name which allows Tethys to distinguish between sources when using multiple sources. For some import maps, the source name will be populated automatically and

should not be changed. If a source name is not populated, in most cases it is acceptable to leave it blank as long as there is not more than one source.

Some data sources, such as most database (e.g., Oracle, MySQL, and PostgreSQL) are hosted on the network, an example of which is shown in Figure 2. In this case, one needs to specify the name of the database, the server on which the database resides (which may be different than the Tethys server), the port (this is an address to which the database listens, and should be populated correctly unless your administrator has configured your database in a non-standard manner), the account username and password.

Other data sources are file based such as Microsoft Excel require a file to be uploaded (e.g., Figure 3). One can either click on the select box to choose a file or drag and drop files onto the dotted target region. Note that if you miss the target, your browser may try to open the file in a new tab which is probably not your intention.

Important note: Some types of documents such as detections allow uploading additional data such as images or short audio segments. These must be selected at the same time as the source document and can only be done with file sources. Multiple files and directories can only be added via the drag and drop interface and must be selected all at once. This restriction may be removed in future releases. If auxiliary information fields are populated in the document and the accompanying files are not included, an error will result. In previous releases, some import tools would attempt to find these automatically. Browser security protocols prevent this with the web-based import tool.

2.1.3 Uploading the data

Once everything has been specified, press the "Upload to Tethys" button. You will hopefully be rewarded with a message indicating that the data have been uploaded successfully. If not, an error message will describe the issue. Note that by default we do not overwrite existing data unless the user checks the "Overwrite existing" data box.

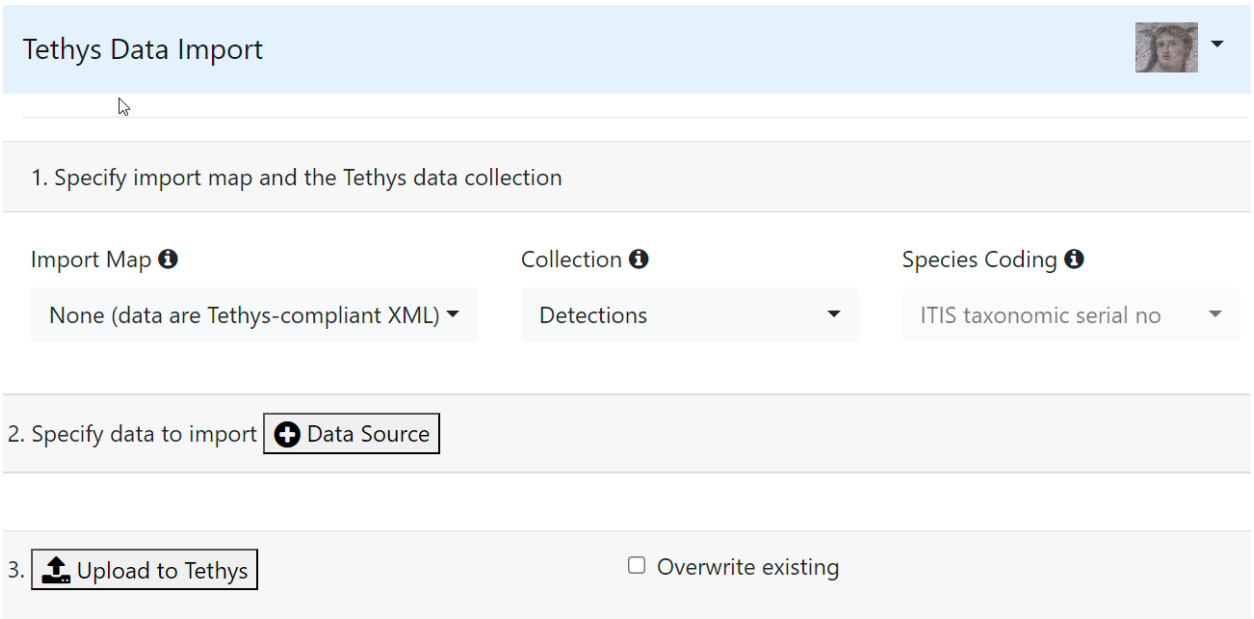

Figure 1 – Web-based import.

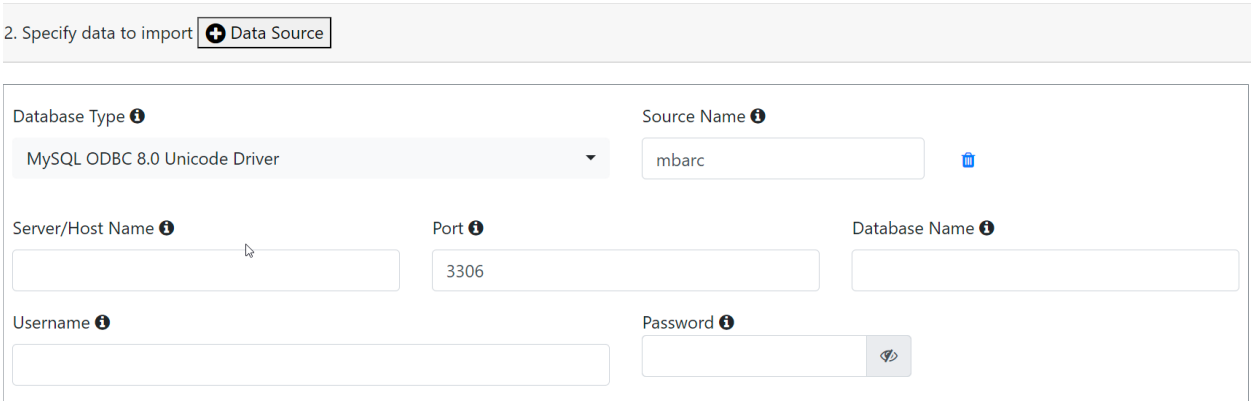

Figure 2 – Example of importing data from a networked database resource.

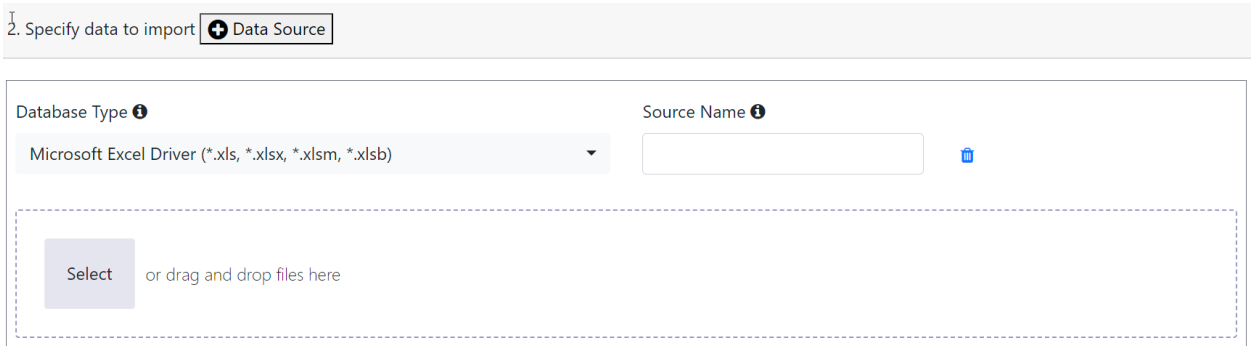

Figure 3 – Example of importing data from a file.

2.2 DBSUBMIT (MATLAB)

A graphical user interface is also provided for document submission by the Java client, although it is typically executed from the MATLAB client. Submission can be done via a single-file resource, an ODBC connection, or a multiple-file resource.

An interface for submitting documents to the database can be accessed from MATLAB with:

```
dbSubmit();
```

To use a specific server, use:

```
dbSubmit('Server','ServerName');
```

The following window should appear:

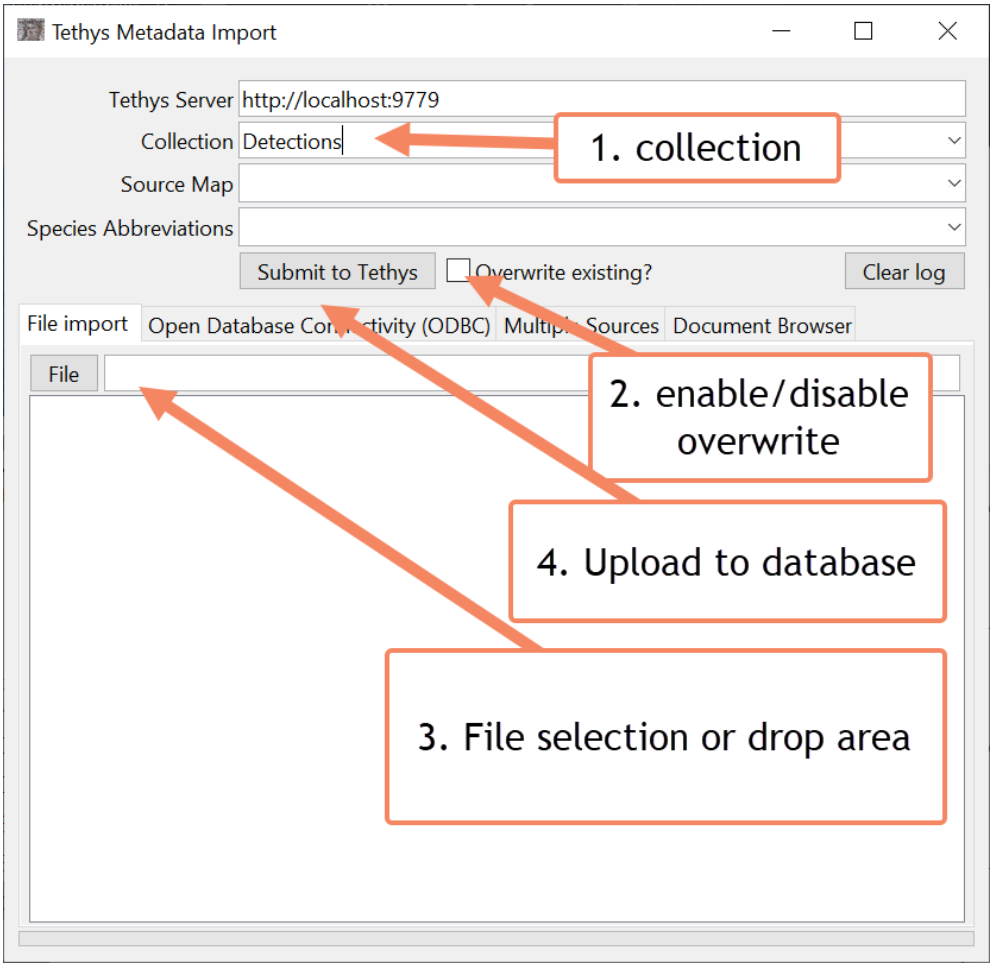

Figure 4 – Data submission interface. The user specifies the Tethys server and collection to which data will be submitted. Source map specifies data field name translation (see section 3.2.1) and when species names are being provided, the user may specify a set of local abbreviations that are used. Tabs indicate where the imported data will be taken from (see text for details).

In the Tethys Metadata Import window, the first input is your server address. In many cases, this will be your local host address which can be written as http://localhost:9779 or http://127.0.0.1:9779.

The second input is a drop-down to choose the appropriate collection to submit your document to (e.g., Detections, Calibrations, Deployments, etc.)

The third input is a drop-down to indicate the appropriate source map. Source maps provide directions on how data contained in the document you are submitting are mapped to Tethys when your data are not already in Tethys-ready XML format. **The Source Map needs to be present on the Tethys server. If you are creating a new Source Map either with a text editor or the source map data import tool, you will need to add the new map to the Tethys server SourceMaps collection before trying to upload data.**

The fourth input is a drop-down to indicate the appropriate species abbreviations to use. Species abbreviations are discussed in section **Error! Reference source not found.** and allow users to specify codes that identify specific species. When the species map is set, the import system will assume that species are denoted using the codings in the species map. For example, the SIO.SWAL.v1 species map uses coding UO to denote detections of odontocetes when the specific species is unknown.

Note that if the Source Map and Species Abbreviation fields can be left blank if you are importing an Excel workbook that already contains a sheet called MetaData that has columns specifying the SourceMap and SpeciesAbbreviation.

Next, there are several tabs to select the submission style:

- File import – Used when submitting one or more files where each file will be submitted to Tethys as a separate document. Spreadsheets and comma-separated file values may be used.
- Open Database Connectivity – Interface for asking the Tethys server to use Microsoft open database connectivity (ODBC) technology to access foreign databases. Users can specify files (similarly to file import) or connect to a networked resource such as a database server.
- Multiple Sources – On occasion, multiple files may be needed to construct a document such as for a deployment that has a separate trackline file. This tab permits multiple document sources to be included.
- Document Browser – This pane allows for a tree view of submitted XML documents by name. Simply specify the collection of the target document, enter its name, and hit Retrieve Document. XML elements can be expanded, and their values examined.

File import will be described here, but further detail on the other import styles can be found in Appendix **Error! Reference source not found.**. To add an individual file from a network location, click on the "File import " button and navigate to the file to be added to Tethys. Click the "Submit to Tethys" button and your document will be submitted, with confirmation (Figure 5) or errors displayed in the message areas.

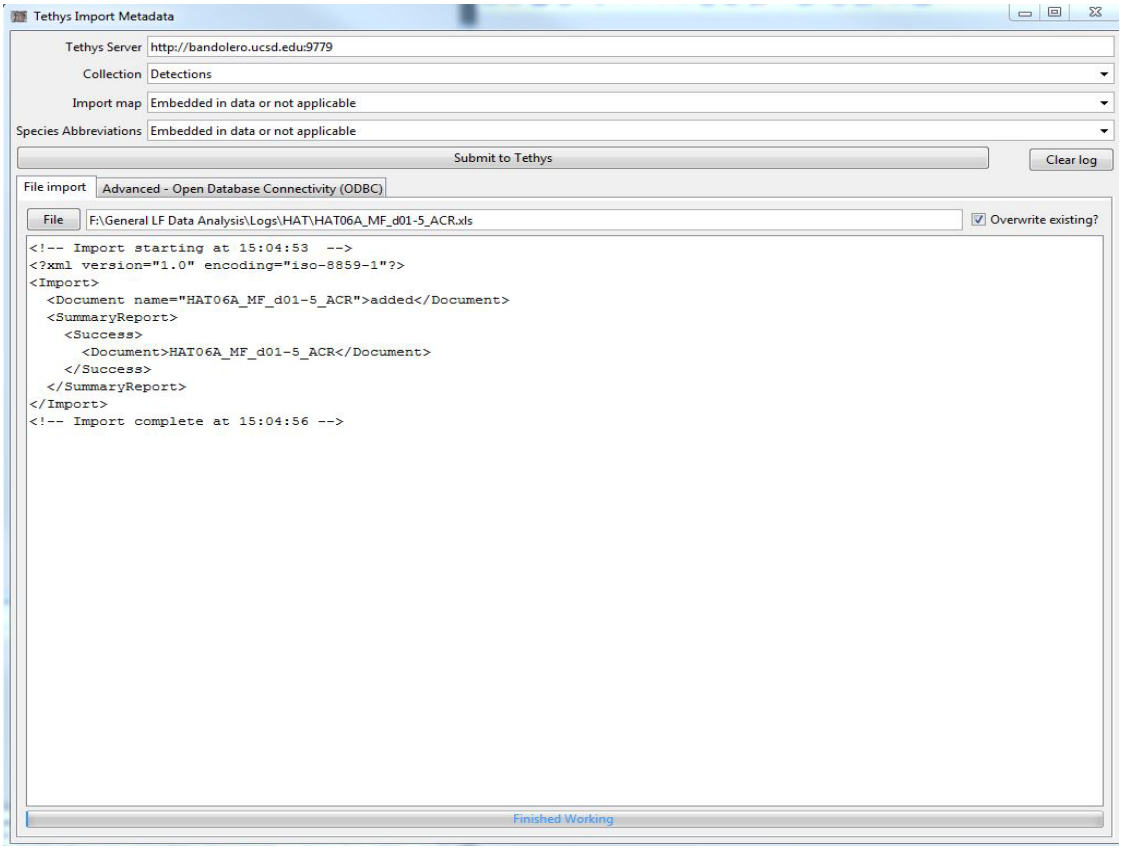

Figure 5 – Successful document submission message.

If you wish to overwrite an existing Tethys document, click the overwrite existing checkbox. Otherwise, trying to submit a document twice will fail and you will receive an error message:

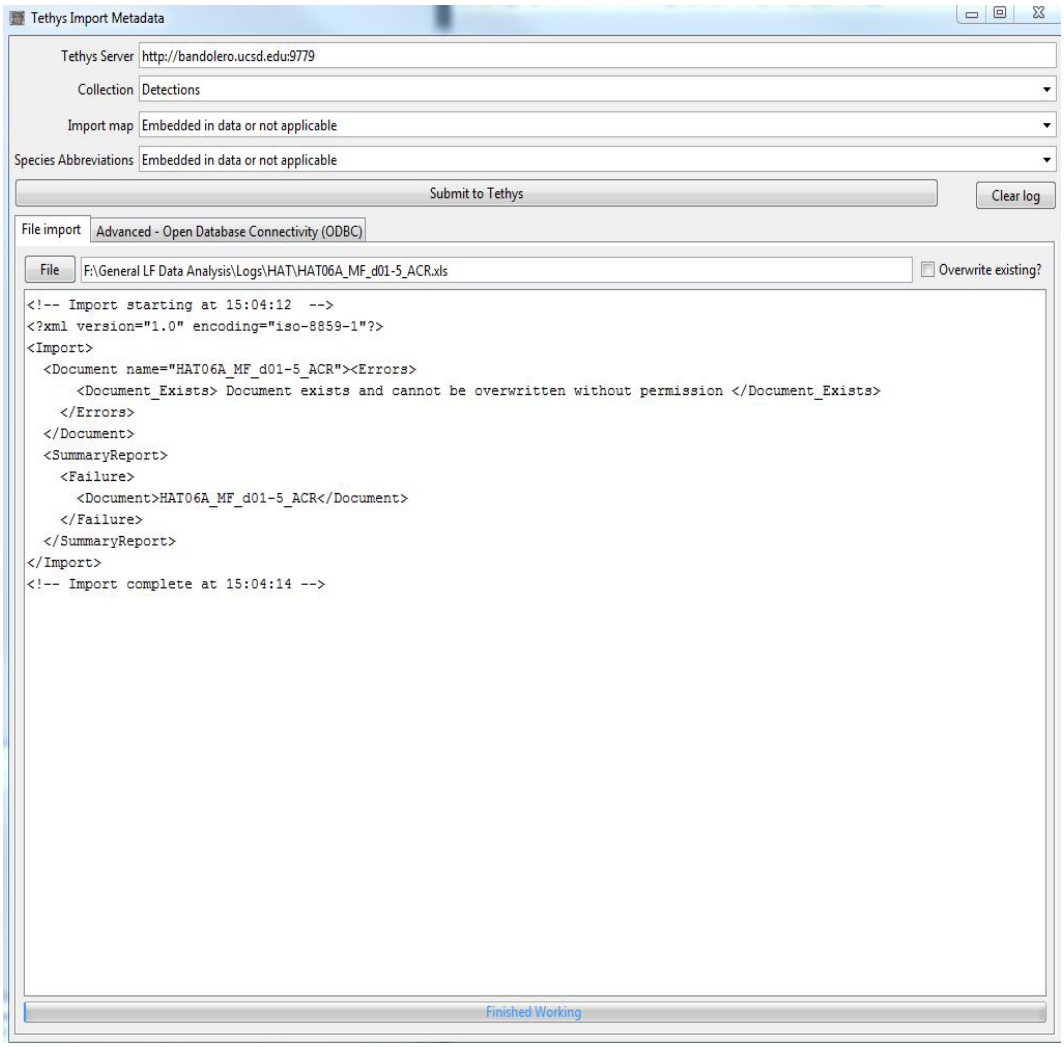

Figure 6 – Document submission error message when a document with the same filename already exists on the database.

More complicated submissions such as connections to databases can be handled on the Open DatabaseConnectivity tab. It has two subtabs, one for file submissions and the other for networked databases. The file access is very similar to the regular file import. The "ODBC networked resource access" sub-tab . Here you can use the drop-down menu to select what sort of database you're trying to connect to (MySQL, MSAccess) which will provide a template for the connection string that ODBC will use to connect. The ODBC option allows for any string to be used. To omit the password from the string and use the encrypted field instead, use Password=<*Password*> in the Connection String, and enter the password in the field above. Note that the web-based import provides a more user-friendly method to do this.

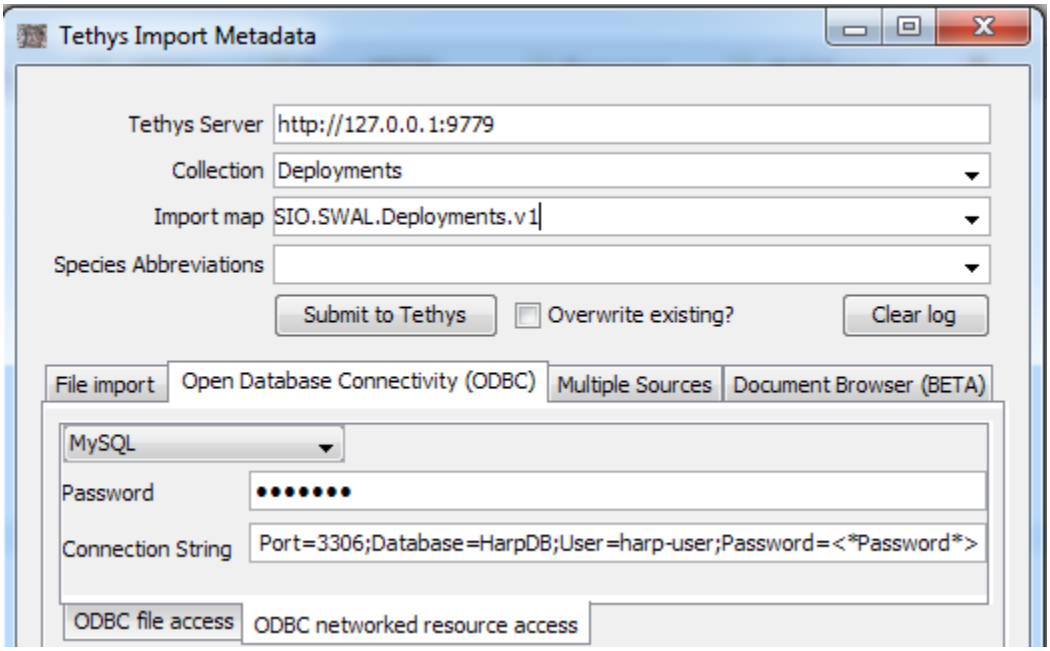

*Figure 7 – ODBC data import using the MATLAB client interface. The password is supplied in the encrypted field by using <*Password*> in the connection string.*

Finally, dbSubmit support multiple sources for file inputs. The Mulitple Sources tab (Figure 8) permits one to specify different files and source names. It is assumed that the source names are specified in the selected import map.

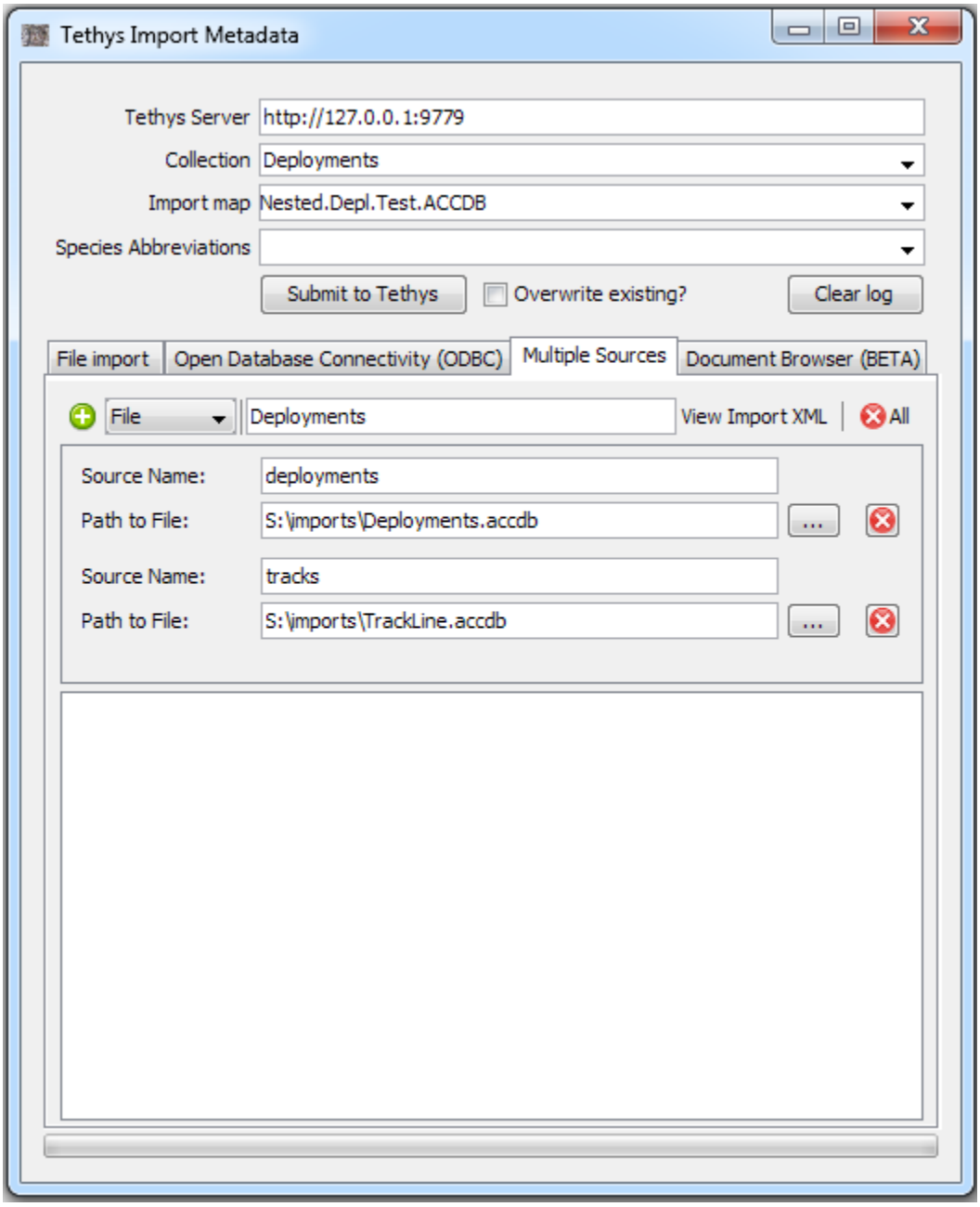

Figure 8 – Data import for multiple sources using the MATLAB client interface.

2.3 PYTHON IMPORT

The Python client's import module permits importing as well. It is intended for programmers and is bare-bones without a graphical user interface. It accepts command line arguments that specify the same type of information about the import as was specified in the graphical data submission tool.

Here is an example of a command line ODBC import into the Deployment container:

```
import.py --sourcemap SIO.SWAL.Deployments.v1 --server tethys.my.org --ConnectionString
"Driver={MySQL ODBC 5.1 Driver};Server=harp.my.org; Port=3306;Database=HarpDB;User=harp-
user;Password=<*Password*>" Deployments
```

The MySQL database is running on machine harp.my.org on port 3306, and Tethys is running on tethys.my.org. MySQL will be accessed with the account harp-user. Setting Password to <*Password*> will result in `import.py` prompting for a password. The ODBC driver is specified in {curly braces} and if not supplied, will default to the above. Data will be imported to the Deployments collection from the database HarpDB using the SIO.SWAL.Deployments.v1 source map.

3 MODIFYING AND REMOVING DATA FROM TETHYS

3.1 DOCUMENT REMOVAL

Documents can be removed by specifying the collection from which they came (e.g., Deployments or Localizations) and the document's name. Note that a document name is not the same thing as the Id field that is used in many collection types, although they are frequently the same. Document names can be seen by producing a document listing for the collection when using any of Tethys's web pages (select from the dropdown with the Tethys goddess image).

Alternatively, for those comfortable with using XQuery, the function base-uri can be used on a document to produce will produce a list of names that includes the collection. For example, the XQuery:

```
for $d in collection("Detections")
return
  base-uri($d)
```

returns a list with entries such as: `dbxml:///Detections/SOCAL32M_odontocete`. In this case, SOCAL32M_odontocete is the document identifier. Standard XQuery functions to filter the results can be used, and if you would like to generate an XQuery, the advanced query page can help you do this without knowing the XQuery language; just modify the return clause to take the base-uri of the documents.

Once the collection and document have been identified, Tethys can to be told to delete the document. The JavaClient library has a function called removeDocument that takes two arguments, a collection name and document identifier which will remove the document. If you are using another language that uses the JavaClient (e.g., Matlab, R), you can simply use removeDocument on the Java client interface (e.g. the query handler returned by dbInit in Matlab: q = dbInit(…); q.removeDocument("Detections", "SOCAL32M_odontocete"). Note that some language clients may provide additional mechanisms such as the dbRemoveDocument method in Matlab, but these will ultimately call the Java Client.

One notable exception is the PythonClient that has its own remove.py script that takes command line arguments --server, --port followed by the collection name and document name. The PythonClient also has a special command called clear_documents.py that can remove all documents from a collection. This should not be used without considerable forethought.

3.2 DOCUMENT MODIFICATION

We do not encourage modifying existing data by modifying the database although this is possible using the XQuery replace mechanism. Rather, modify the document that was submitted and import it again with overwrite enabled.

The rationale for this is that a copy of each source document submitted to Tethys is saved in addition to submitting it to the database. The documents are stored in the subfolder of the database's *source-docs* folder[1], which corresponds to the collection name. This makes it possible to rebuild the entire database should there be a case of catastrophic failure. Modifying the database directly would mean that the source documents would no longer synchronized with the database contents and such rebuild would no longer be possible.

3.2.1 Adding documents that have been accidentally removed

As a copy of documents have been saved, these copies can be used to regenerate data that have been removed. In general, you should not need to rebuild a collection. In the history of this project, we have never seen a database corrupted except in the case of disk failure (back up your data). Should you need to rebuild a collection from the source documents that are saved, use the Python client program `update_documents.py`. The following is an example where any document that was inadvertently removed from SourceMaps is added back in provided that it has a copy in sourcedocs (remember documents from databases are not copied to sourcedocs). Here, `pythonclient_path` refers to the path to your *PythonClient* folder and `python_path` refers to the path to the folder that contains the Python that was distributed with Tethys:

```
current_dir> cd pythonclient_path
REM If pythonclient_path is on another drive, current_dir will not change
REM and you must take the additional step of changing drive letters
REM e.g. pythonclient_path is on drive e, uncomment the next line:
REM current_dir> e:
pythonclient_path> python_path\python.exe update_documents.py SourceMaps
```

In general, any number of collections can be listed to be updated. To update all collections, use the keyword all:

```
pythonclient_path> python_path\python.exe update_documents.py all
```

If you wish existing documents to be overwritten by their copy in sourcedocs, use the `--update=true` flag. This is not usually needed unless one has done a global search-and-replace across source documents and wishes to incorporate the modified documents into the database. Finally, the optional `--clear` flag will have the same effect as running the `clear_documents.py` command prior to `update_documents.py`).

4 IMPORT MAPS

Import maps do not directly import data. Rather, they are a recipe for how to organize and transform data such that they can be stored in a Tethys database. Once the import map is established, it can be

[1] No records are written when the source material comes from a database as it is assumed that the database itself is backed up.

applied repeatedly to multiple instances of your data. We support two methods to construct data import maps. The first is via a web interface that allows you to interactively associate your data with the standardized names used by Tethys. The second is a textual specification that can be created in a text editor and uses "markup" annotations to specify how data are translated. The web interface ultimately generates a markup annotation that is stored on the Tethys server. Once an import map has been specified, it can be used to load any data in the same format as the specification into Tethys.

We can map any type of data for which there is an open database connectivity (ODBC) driver. ODBC drivers allow many types of files to be treated as if they were sequential query language (SQL) databases including comma separated value (CSV) files, Excel workbooks, Open Office Calc documents, and of course, most databases. It is necessary for the Tethys server to have an ODBC driver installed for the type of data to which you wish to connect. Your Tethys server administrator or your information technology staff should be able to help you install any additional needed ODBC drivers which will require administrator privileges to install. There is information on installing some common ODBC drivers in the main Tethys manual.

4.1 PLANNING DATA IMPORT VIA THE WEB INTERFACE

The easiest way to create an import map is to visit your Tethys web server page (e.g., mytethysserver.noaa.gov:9779/Client) and to select "Plan data import" from the dropdown menu with the Tethys logo in the upper right corner. This will navigate to the import map planning tool (

Figure 9). The upper-most row of the display is used for opening an example of the data that you plan to import, specifying which Tethys data collection you will be adding to (e.g. a deployment or set of detections), saving and loading drafts of your specification, and publishing the source map to Tethys once you have specified your mapping. The icon ⓘ is an information symbol that appears throughout the page. Clicking on ⓘ will display additional information about how to use the program.

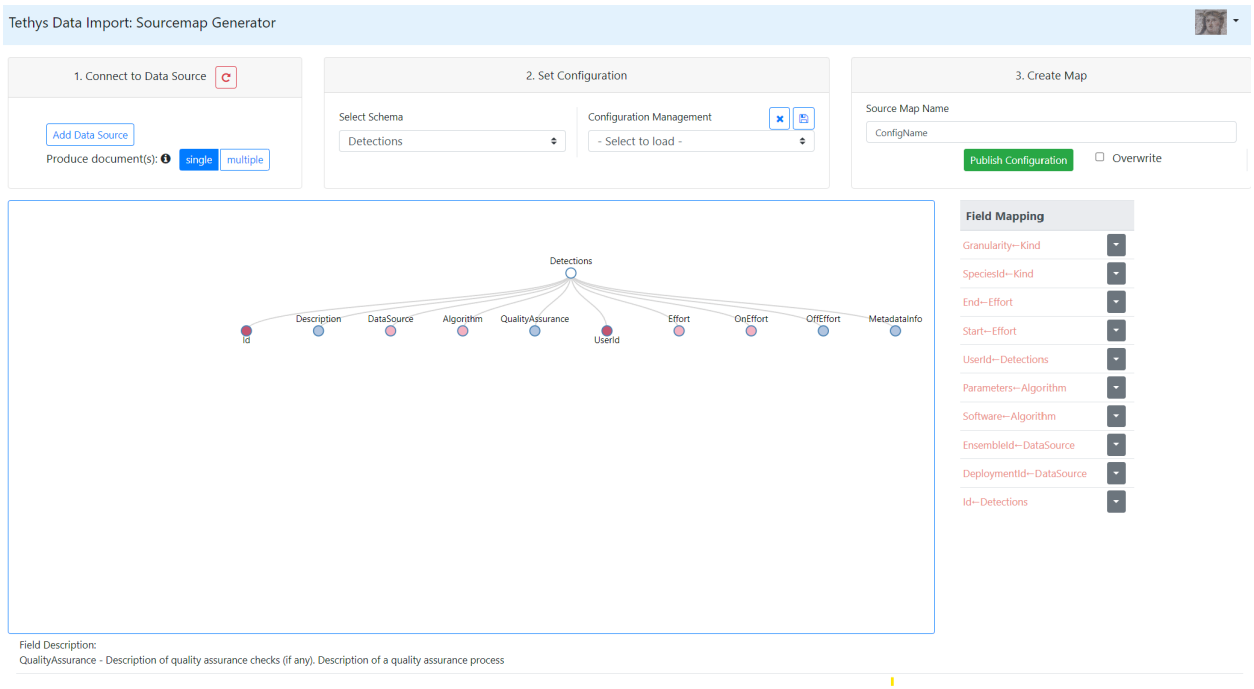

Figure 9 – Import map planning tool. When first opened, the tool is not associated with any of your data.

4.1.1 Connecting to data

Your first step is to connect to an example of your data source. This lets you create a plan for importing data based on an example. The "Add Data Source" button in the top left will open a dialog box (Figure 10) that lets the import map tool connect to an example of the type of data for which you are interested in constructing a source map. The fields of the dialog will change depending on the type of source to which you wish to connect. The following fields are always present:

- Database type – To what type of data do you wish to connect? The values present here depend on which ODBC drivers have been installed on the Tethys server machine (see the ODBC discussion, p. 5). Some drivers have variants that can be confusing. When given the choice between Unicode and ANSI drivers, select Unicode which specifies how letters and numbers are stored in the computer. If drivers are marked x64 and x32, or x64 and one without the x64, pick x64 as this denotes a 64 bit operating system which is what your Tethys server will be running on.

  Some common drivers that might appear:

  - Microsoft Access Text Driver – This is used to process comma separated value (CSV) files. When this option is present, each .txt or .csv file is assumed to have a header row and each file is treated as if it were a table named after the text file in which data are stored.
  - Microsoft Access Driver – This is used for the Microsoft Access database. We assume that your Access database is not networked (most are not) and you will submit a copy of your Access database via a file dialog.

    - Microsoft Excel Driver – Permits connections to Microsoft Excel workbooks. Column headers are expected, and each sheet is displayed as a table. Due to peculiarity of the Microsoft Excel driver, these sheets will have a dollar sign ($) appended to their names.
    - MySQL– Access an open source MySQL database server.
    - PostgreSQL - Access an open source Postgres server.
- Source Name – This is a label that can be used to distinguish between tables with the same name when multiple sources are used. If you add additional sources, their names must be unique.
- Rows displayed – The source map generator will display a few rows of your data once you have made the connection. This value is the maximum number of rows that will be shown. You should not typically need to change this.

If your source is in a file such as an Excel workbook, the dialog will have a relatively few number

Figure 10 – Connecting to a data source dialog. The Add Data Source button presents this dialog which will display an example of your data after it has been successfully completed.

In the dialog of Figure 10, to add an Excel Workbook one would only need to attach an appropriate workbook and specify the sourcename before clicking on the connect button. Additional fields may appear for other types of database. For example, when connecting to a database that serves over the network such as MySQL, PostgreSQL, and Oracle Database, one would see fields prompting for the following:

- Server/Host name – The name or internet protocol (IP) address of the machine hosting the database. This is not necessarily the same machine as the Tethys server.
- Port – The port (essentially a mailbox number) on the server machine that will be servicing queries. Unless the database administrator has changed the port to a non-standard one, you are likely to be able to connect without changing this from its default.

- Database name – The name of the database to which you wish to connect. If you are not sure what should be used here, contact the person who supports the database to which you wish to connect.
- Username – A user account name that is authorized to query the database.
- Password – The password associated with the username. Security note: This password is transmitted to establish the connection. If you are running Tethys as a secure server, the password will be encrypted and secure. Running Tethys in secure server mode requires a certificate (see the Tethys manual), and many Tethys users choose not to take this step. If you are operating in a corporate/government environment behind a firewall or on a virtual private network (VPN), this may be acceptable. Do not enter passwords in unsecured environments such as a coffee shop network if the Tethys server has not been started using a secure protocol. If you connected with an https address, you connected securely whereas http is not secure. Web browsers typically display a symbol (e.g., a lock) to indicate when your connection is secure.

Once you have connected, the interface will show the tables associated with your entry.

4.1.2 Understanding the display

The middle and lower sections of the display consist of three regions (Figure 11).

- The Tethys tree shows the structure of the Tethys data. Fields with a light blue or light pink coloring are fields that will not contain data, but consist of subtrees with further information. Color is used to denote function. A deep red circle indicates a mandatory field that has not yet been associated with data or a default value. White circles indicate optional fields. Pink and blue circles indicate that there are fields in a subtree. These can be expanded (or collapsed) by clicking once on the white circle. Note that some fields are optional but have children that are marked as mandatory. An example of this is the MetadataInfo field that is present in most of the Tethys data schemata. MetadataInfo can be ignored even though it has mandatory children, but once one mandatory child is associated with your data source, the other mandatory children must also be mapped to the data source. Moving your cursor over an element will update the field description that is shown underneath the tree. The mouse wheel controls zoom and clicking and holding anywhere there is not a field permits you to move the entire tree around.
- The tables associated with the source data can be seen at the bottom of the display. One can click on each tab to see the contents of the table. When there are too many tables to display, these become scrollable. Specific tables can be easily found with the search box that is just above the tables.
- The final area contains a list of field mappings that show which required Tethys fields have not yet been mapped to user data (light red) and a list of elements that have been mapped (forest green). One can think of paths from the root of the Tethys tree to an element that will be mapped. For example, representing duty cycled recordings in the Deployments schema is accomplished by several elements in the the Deployment→SamplingDetails →Channel → DutyCycle → Regimen element: TimeStamp which indicates when the duty cycling regimen started, RecordingDuration_s which is the number of seconds that we record at the start of a duty cycle, and RecordingInterval_s which is the amount of time between consecutive starts of

recording (time not spent recording is RecordingInterval_s – RecordingDuration_s). In general, the field mapping lists the last two fields along the path in reverse order. So, one might see: RecordingInterval_s ← Regimen. If the field has been bound to an entry in one of the tables, this will be followed by a right arrow and the name of the field to which it is bound. If RecordingInterval_s were bound to RecordingInterval_m, the following would appear in the list:

RecordingInterval_s ← Regimen → RecordingInterval_m

The list can be alphabetized by clicking on the Field Mapping title bar. Buttons allow manipulation of the items in the list and are described below along with the drag and drop targets default and condition.

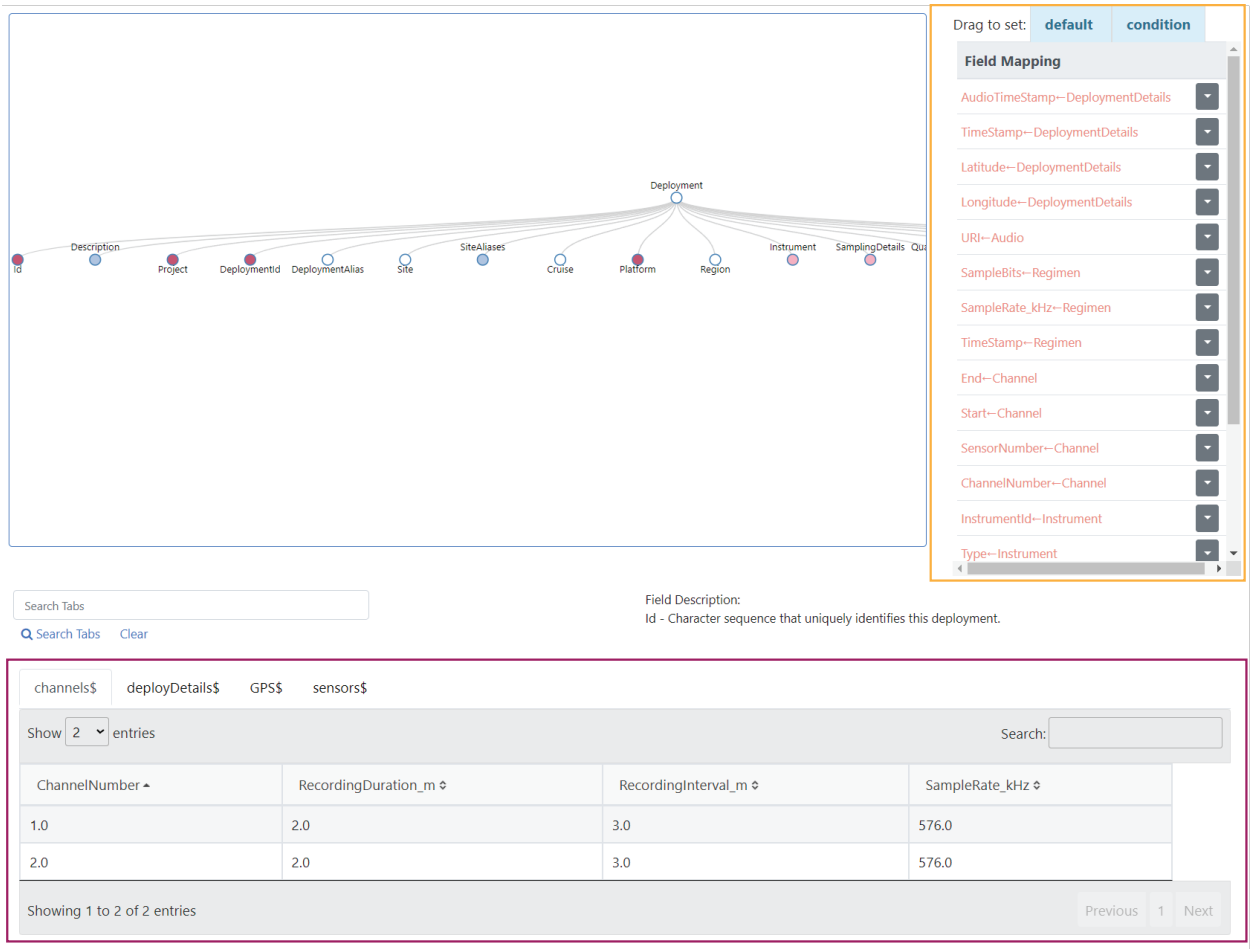

Figure 11 – Middle and lower sections of the import map generator. Boxes have been added to highlight different regions. The blue box contains Tethys data elements. The red box shows example data from the data sources that have been connected. The orange box shows a list of Tethys fields and the table entries to which they are mapped. Color is used to denote mandatory fields and those that have already been associated with a data source attribute.

4.1.3 Associating Tethys and source table fields

Tethys fields may be double clicked. Doing so creates a field name that can be dragged onto the column header of any table in the bottom section. When this is done, the field mapping list is updated, and the Tethys field is colored green.

Multiple Tethys fields may be dragged to the same Tethys field. In most cases when this is done, you will be asked if you wish to replace the mapping with the new field or if you wish to add it to the existing one (Figure 12). The default is to replace the mapping, but adding a new one permits you to combine two fields, either textually or as a calculated value. Should you choose to combine the fields, you will see the field names listed in square brackets [ ] listed as a three items separated by colons: [source-name:table-name:table-column]. You can rearrange the text as you see fit, for example, adding text in between fields or performing numerical operations such as converting to a different unit if you mark the combination as a calculation.

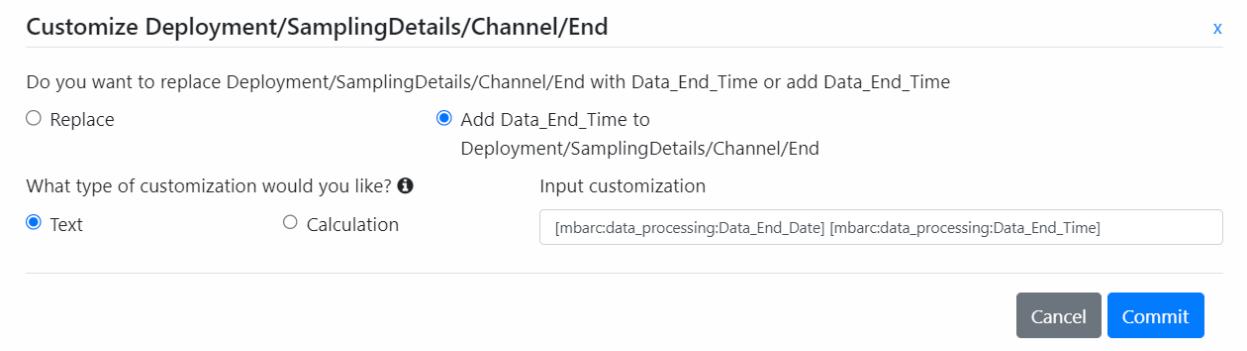

Figure 12 – Dragging a Tethys field to a second table field asks if you wish to replace the mapping or add to the existing one.

4.1.3.1 Values that are not in the schema

Some fields of the Tethys schemata are designed to be extensible (e.g., user defined parameters). When this is the case, the field can be dragged to multiple table fields. In this case, the source map will map the user data to the same field name as the column name. The column name must be a valid XML element name (e.g. no spaces, punctuation, etc.). This limitation may be removed in the future.

4.1.3.2 Default values

To provide a default value for a Tethys field, double click on it, drag to the blue default target above the field mapping (Figure 13), and click again. You will be prompted for a value. If the Tethys field is mapped to a table field, the default value will be used when the table field is empty. If it is unmapped, the specified default will always be used. Entries with default values are denoted in the table with the letter [D] as shown in Figure 13. If a field is not associated with the item, the default value will be shown.

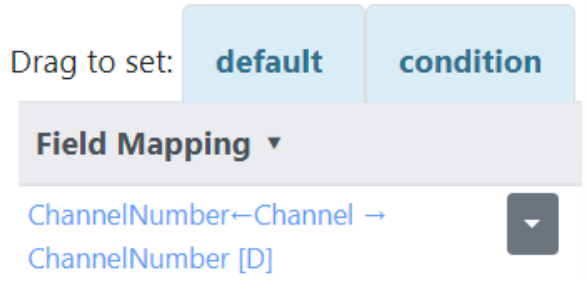

Figure 13 - Default fields and conditions. Tethys elements may be dragged on top of either of these two target. Double click on the Tethys element, drag it to one of the two fields above, and click again to drop. Note that the drag to set fields do not appear until after at least one data source has been specified.

4.1.3.3 Conditions

Conditions permit entire sections of the Tethys tree to be omitted when certain conditions occur. For instance, if our instrument database has empty values for the duty cycle fields when an instrument was not duty cycled, we can add a condition such that these entries will only be produced when the field is present. A condition can be set by dragging a Tethys field onto the word condition above the field mapping (Figure 13) or by using the dropdown menu on . A new dialog will appear (Figure 14). Field map items with conditions have a [C] appended to them. For numerical fields, the

Figure 14 – Adding a condition. When the Recording_Interval_min field in table deployment_information from a source that has been named "mbarc" is blank, the entire DutyCycle section will be omitted.

4.1.3.4 User Definable Fields

Some fields in Tethys can have arbitrary values associated with them. These fields are denoted by a dashed line around the circle (Figure 15). This field can be dragged multiple times to tables in the example data. When data are imported into Tethys, each of the associated user data fields will appear as a user-defined name in the Tethys schema. This allows users to extend the Tethys schemata with their own definitions. As an example, if we dragged parameters to a pair of user fields: Threshold and AveragingWindow_ms, these would both appear as children of the Parameters field when data are imported and could be queried like any other Tethys parameter. Some user fields may not be valid XML names which must start with a letter and consist of letters, digits, underscores (_), dashes (-), and periods (.).

Advanced users may wish to provide structuring of their additional data (e.g., place all things affecting signal processing in a group) this is not possible with the GUI tool, but the published import map can be modified to accomplish this.

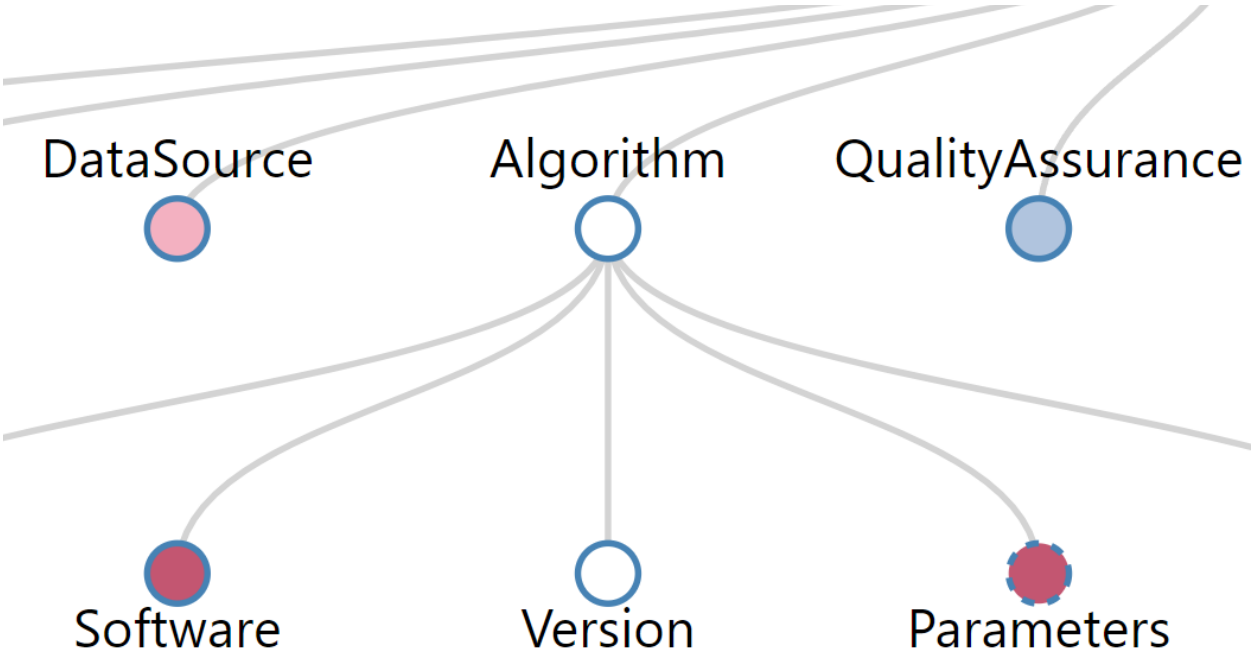

Figure 15 – Example of a Tethys field that will accept any number of user defined fields. The Parameters field has a dashed line rather than a solid one. It can be dragged to any table element multiple times and each of those will appear as a child of Parameters in the XML produced by the import map.

4.1.4 The field map dropdown

Tethys fields that have been associated with user data fields, default values, or conditions have entries in the field mapping list (Figure 11, orange box). Each entry has a drop down menu. The available menu choices can vary depending on how the current status of the entry:

- Find – Moves the Tethys tree such that the field is visible.
- Condition – Add or edit the condition associated with the Tethys field.
- Default value – Add or edit a default value associated with the Tethys field.
- Delete – Remove all associations with this Tethys field
- Filter – Restrict the query such that only values meeting a user-specified criteria are included (see below).

4.1.5 Saving and publishing the import map

Drafts may be saved, loaded, and erase from the configuration manager at the top of the window (Figure 9, p. 17). Pressing the floppy disk icon will save the configuration as the current Source Map name (in box 3). Selecting a configuration from the dropdown menu will load a new configuration. In most cases, the source will be partially loaded (displayed in red) and will need to be edited. Pressing x will remove the current draft.

Once everything has been set up properly, we recommend that you save a draft as it will let you make modifications to the import map without having to directly modify the generated XML file.

Pressing the publish button will generate a new Source Map and upload it to the server. In some cases, queries to multiple tables will be generated on your behalf. With a well-designed database, the server can often determine how these tables fit together. However, if this has not been provided by the database manager, or if the data source is a CSV file or Excel workbook, you will need to do this manually. In such cases, a dialog will list the tables that are being used for each query and will ask you to provide criteria to link them together (Figure 16).

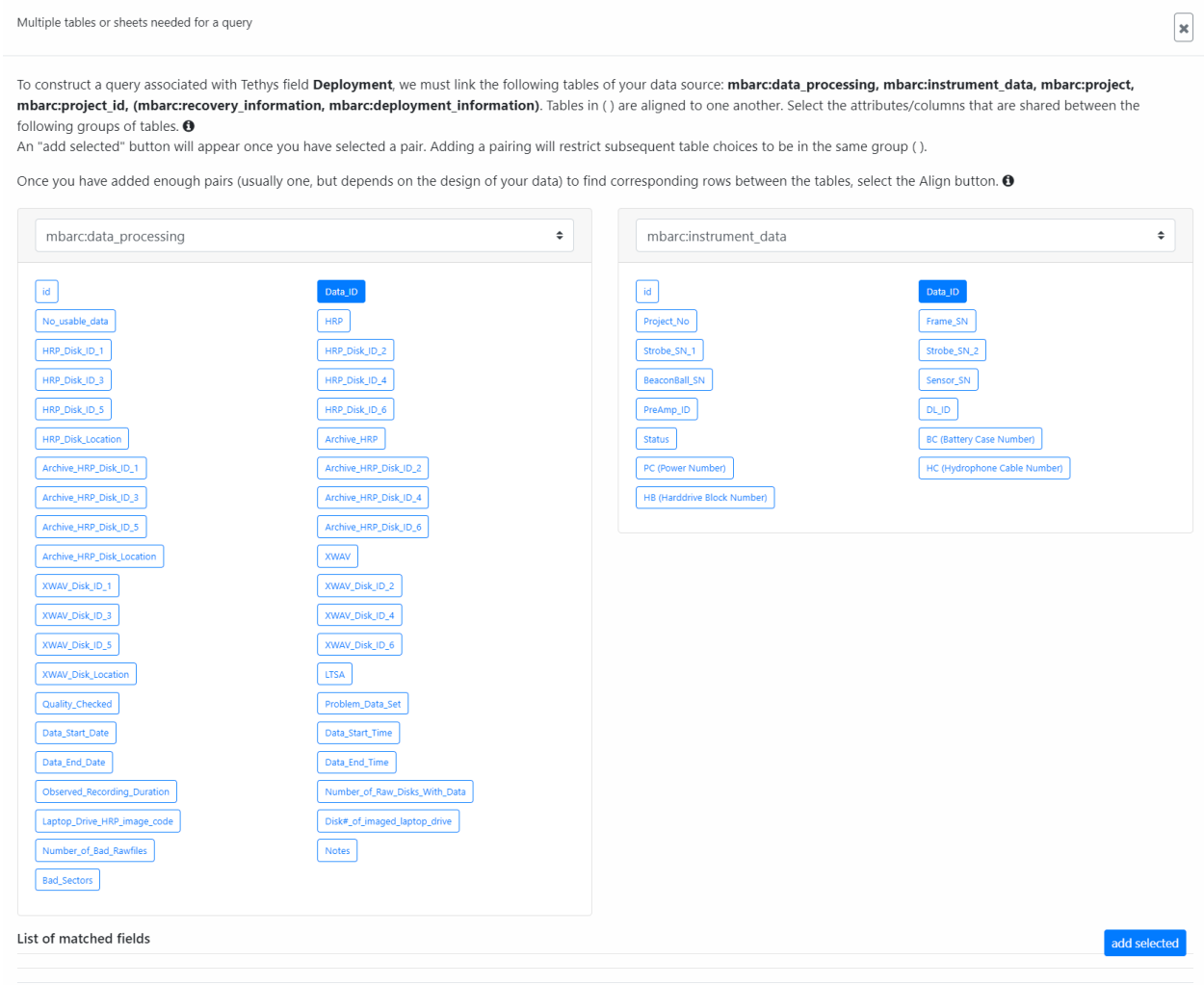

Figure 16 – Example of linking tables during publication.

In the example (Figure 16), a SQL query is being created at the top level of the Tethys element tree for deployments. In this case, 6 tables are being used in this query. Two of the tables, deployment & recovery information, have already been linked by the database manager. Consequently, we need to associate 5 different groups of tables. The left and right column each show one of the 6 tables and can be changed. In this database, Data_ID serves as a key to align different tables. That is, if we want to keep rows of data_processing and instrument_data aligned with one another, we can do it by matching the Data_ID in each table. There is no need for these to have the same name, but in this case they do.

When the appropriate fields have been matched, click on add selected. If more than one pair of fields is needed to align the tables, additional fields may be selected and added. After one pair has been

specified, an additional button “Align tables” will appear. Clicking will merge the pair of tables into a group, indicating that we now know how to align them in a query. You must continue to repeat this until all tables are aligned. If additional Tethys fields need alignment, the process will start again with the new field. Once this is completed, the source map will be published.

4.2 PLANNING DATA IMPORT BY SPECIFYING AN XML FILE

The previous section showed how to generate a data import specification using a web-based data import planning tool. When one publishes the data import plan, an XML file is generated. Alternatively, data import can also be specified by an XML file. In most cases, using the web-based tool is preferable, but the XML file provides some additional controls that are not yet supported in the web tool such as converting the case of a field to ALL CAPITALS or all lower case. In such cases, users may wish to write their own XML file or modify one created by the web-based tool.

As discussed in sections **Error! Reference source not found.** and **Error! Reference source not found.**, data can be imported into Tethys in a variety of formats. Sources can include comma-separated value files, spreadsheet workbooks, databases, and anything else that supports the open database connectivity protocol. When the data being imported is not already stored in XML, a translator, called a source map, needs to be used to map the row-oriented data to XML.

The following sections provide additional details and examples for creating source maps and the different methods of importing data into Tethys.

4.2.1 Source Maps

A source map is an XML document that provides detailed information on how to translate from a non-XML data source to the Tethys XML schemata. In general, a source map will be created to import a group of documents that have the same structure. For example, a program might produce spreadsheets, text files, or database records describing soundscape measurements. A source map would be established for this program that could then be used to import soundscape measurements produced by the program for multiple recordings.

As introduced in section **Error! Reference source not found.**, source map documents have the following structure:

```
<Mapping>
	<Name>MyMap</Name>
	<DocumentAttributes>
		List of <Attribute> elements that will be applied to the document
	</DocumentAttributes>

	<DocumentSources>
	    Optional list specifying what type of data sources are expected.  Populating this
	    section will make it easier for users to submit data as it will enable automatic
	    population of fields for some of the import clients.
	</DocumentSources>

	<Directives>
	    Elements that provide document structure and specify how fields in the source
	    spreadsheet, database, etc. correspond to elements in the resulting XML document.
	</Directives>
</Mapping>
```

The value of the <Name> attribute must be unique and is used to identify the source map. The <DocumentAttributes> define the XML namespace. In most cases, one will simply copy the following Attributes:

```
  <DocumentAttributes>
    <Attribute>
      <Name>xmlns</Name>
      <Value>http://tethys.sdsu.edu/schema/1.0</Value>
    </Attribute>
    <Attribute>
      <Name>xmlns:xsi</Name>
      <Value>http://www.w3.org/2001/XMLSchema-instance</Value>
    </Attribute>
    <Attribute>
      <Name>xsi:schemaLocation</Name>
      <Value>http://tethys.sdsu.edu/schema/1.0 tethys.xsd</Value>
    </Attribute>
  </DocumentAttributes>
```

Each attribute <Name>/<Value> pair will be appended to the first element of the resulting document. As an example, if the map creates a Detections document, the above <DocumentAttributes> will create the following:

```
<Detections xmlns="http://tethys.sdsu.edu/schema/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://tethys.sdsu.edu/schema/1.0 tethys.xsd">
        …
</Detections>
```

The <Attribute> section may be followed by an optional section <DocumentSources> that permits one to specify the types of documents that will be imported. In some scenarios, users may wish for multiple data sources (e.g., two Excel workbooks) to be used. Each source is contained in a <Source> section that requests a <Name> that is used to distinguish between tables (sheets) of the different sources and a <OpenDatabaseConnectivity> (ODBC) section that specifies how the system should be reading the source. Any ODBC driver installed on the server can be used in this field, and one can use a part of the name. For example, to indicate that we are reading from an Excel Workbook, we might use:

```
  <DocumentSources>
    <Source>
      <Name>pifsc</Name>
      <OpenDatabaseConnectivity>Microsoft Excel</OpenDatabaseConnectivity>
    </Source>
  </DocumentSources>
```

The <Directives> element allows one to actually perform the data translation. Most elements are simply copied. For example:

```
<Directives>
        <Deployment>
                <Project>SOCAL</Project>
                <DeploymentId>34</DeploymentId>
        </Deployment>
</Directives>
```

would produce the following XML document:

```
<Deployment xmlns="http://tethys.sdsu.edu/schema/1.0" [other attributes…]>
       <Project>SOCAL</Project>
       <DeploymentId>34</DeploymentId>
</Deployment>
```

Several special elements documented below can be used to provide mappings of fields between a user's data source and a Tethys document.

4.2.2 Directive and Tables: Accessing the source data

In most cases, we would like the element values to be populated with values from our source document. The `<Table>` element allows you to access relational databases as well as spreadsheets and CSV files. When a `<Table>` element is encountered in the `<Directives>` section, the element `<Entry>` can be introduced to allow data to be mapped from the source document to XML.

`<Table>` requires an attribute, `query`, which will be a database query in sequential query language (SQL). SQL is beyond the scope of this manual, but *SAMS Teach Yourself SQL in 10 Minutes* (Forta, 2016) is a recommended introductory primer. Relational databases group information into tables, and the `<Table>` keyword let us access one or more tables with a SQL query that is specified as an attribute named **query**, an example of which can be seen in the Table example below:

```
<Directives>
    <!-- Note $ at end of Deployments as from a spreadsheet -->
    <Table query="select * from Deployments$;">
       <Deployment>
              <Entry>
                     <Source>[Project Label]</Source>
                     <Dest>Project</Dest>
              </Entry>
              <Entry>
                     <Source>[Project #]</Source>
                     <Dest>DeploymentId</Dest>
              </Entry>
       </Deployment>
    </Table>
</Directives>
```

There are several important things to consider when writing queries using `<Table>`:

- The query string is considered an attribute. It may be enclosed with single or double quotes. Should you need to nest a single or double quote within the string, the XML escape sequences ' and " may be used respectively. Examples

  query = 'select start, end, species from Detections where species = "*Grampus griseus*";'

  query = "select start, end, species from Detections where species = "*Grampus griseus*";"

- Each sheet within a workbook is considered a table and uses the sheet name as the table name. The Microsoft open database connectivity (ODBC) software that accesses the workbook appends a dollar sign ($) to Excel table names. Providing meaningful names to your sheets, e.g.,

hydrophones, as opposed to the default Sheet#, will make your life easier. The first row is considered a column name.

- The Microsoft ODBC software accesses a directory of comma separated value (CSV) files. Each CSV file is considered a table. Hence, data that are to be imported should be in distinctive folders with identical filenames. The filename extension is considered to be part of the table name. Like spreadsheets, it is assumed that the first row contains column names.

The Directives on the previous page show an example designed for importing data from an Excel workbook. Note that this is a partial example for pedagogical purposes. While it will produce valid XML documents, they will not conform to the Tethys schemata expectations.

As `<Table>` is the very first element after `<Directives>`, an XML document will be produced for each row of the workbook's Deployments sheet. For workbooks, the first row must contain column headers with subsequent rows providing data. The `<Source>` element indicates the name of row header and the `<Dest>` element indicates the name to which it will be mapped. Note that the Source element field name is enclosed in square brackets [ ].

If our workbook contained:

Project Label	Project #
SOCAL	32
SOCAL	33

two XML documents would be produced (note that this example does not conform to the Deployment schema as there are required elements missing):

```
<Deployment …>
        <Project>SOCAL</Project>
        <DeploymentId>32</DeploymentId>
</Deployment>
```

and

```
<Deployment …>
        <Project>SOCAL</Project>
        <DeploymentId>33</DeploymentId>
</Deployment>
```

Alternatively, if we change the order of `<Deployment>` and `<Table>`, one document with two entries is produced as `<Table>` is a child of `<Deployment>`. This is not what we want as our schema defines each Deployment as a separate entity, but there are other situations where having multiple "rows" is useful, such as in a Detections or Localizations document.

```
<Deployment …>
        <Project>SOCAL</Project>
        <DeploymentId>32</DeploymentId>
        <Project>SOCAL</Project>
        <DeploymentId>33</DeploymentId>
</Deployment>
```

Table supports optional attributes that are only rarely needed and that most users can ignore:

- source – This is an advanced option that can be used when connecting to multiple data sources (e.g., two databases). When multiple data sources are present, we need to know which one to query against. When submitting data, you will be asked to provide a name for each source. We match the data source name against this list and query the appropriate database.
- label – This is only necessary when you have nested queries that select attributes with the same names. If a label is provided, it can be used in an entry to specify a table attribute that would otherwise be hidden by a nested query. This attribute is primarily provided for legacy purposes. In most cases, a better solution is to distinguish fields by including their table name. For example, if table A and table B have the attribute data_id, one can write `A`.`data_id` and `B`.`data_id` to distinguish them in the queries:

  ```
  <Table query=”select `A`.`id` from `A`;”>
     <!—more directives … -->
     <Table query=”select `B`.`id` from `B` where `A`.`id` = `B`.`id`;”>
        <!—more directives … -->
     </Table>
  </Table>
  ```

  Note that the table names and fields have backquotes around them. In this case, it is not necessary, but doing so allows for special characters in names such as spaces which sometimes occur in databases.
- optional – Sometimes a query is needed for a table that may or may not exist. Use <Table query="some SQL query;” optional="true"> in such cases. Everything enclosed in the <Table> element will only be populated if the table exists.

4.2.3 Sheet (deprecated)

Although `<Table>` is recommended, the legacy directive `<Sheet>` can be used for spreadsheet workbooks and comma-separated value sources. When used with a workbook, the attribute `name` is expected and specifies the name of the workbook page. By default, Excel uses Sheet1 as the name for the first sheet, Sheet2 for the second, etc., but users can rename these. The following `<Directives>` section would be equivalent to the `<Table>` syntax:

```
<Directives>
	<Sheet name="Deployments">
		<Deployment>
			<Entry>
				<Source>[Project Label]</Source>
				<Dest>Project</Dest>
			</Entry>
			<Entry>
				<Source>[Project #]</Source>
				<Dest>DeploymentId</Dest>
			</Entry>
		</Deployment>
	</Sheet>
</Directives>
```

4.2.4 Entry directives

`<Entry>` directives have several other features. Multiple source fields can be combined simply by placing them in the same `<Source>` element.

The `case`, or capitalization, of textual source material can be modified as well by providing an attribute to a `<Source>` element `<Source case="value">` where value is:

- `upper` – all capitals
- `lower` – all lower case
- `capitalized` – the first letter of the first word is capitalized, or
- `title` – the first letter of each word is capitalized.

Capitalization can be important as queries are case-sensitive.

Two additional elements that can be present in an entry are `<Default>` and `<Kind>`. If a `<Default>` element is provided, its value is used when the source field is absent or has no value present. Finally, a `<Kind>` element can be used to specify special formats.

The following are the currently valid `<Kind>` inputs:

DateTime – Date and time from a number of standard formats. When multiple fields are included in the `<Source>` element, they are combined. For instance, if the data source broke a timestamp into day and hour, specifying `<Source>[day][hour]</Source>` could be used to combine the two.

Integer- `<Value>` is converted to an integer.

LatLong – `<Value>` is interpreted as a latitude or longitude. Latitudes and longitudes are stored internally in degrees ([-90,90] south/north and [0, 360] east respectively). The following formats are accepted:

- Number in the range [-90,90] or [0, 360] representing degrees north/south or east. This is the format in which Tethys stores latitudes and longitudes.
- Degrees minutes seconds (DMS) followed by an optional directional indicator: N, S, W, E. Numbers in the DMS can be separated by anything that is not a number, the strings “36 24 02.5 W” and “36° 24’ 2.5 W”, and “36 degrees, 24 minutes 2.5 seconds w” will all be converted to a longitude of approximately 323.5993 degrees east.

Number – `<Value>` is converted to a number.

SpeciesCode – The field is to be treated as a species identifier. This invokes the lookup routines to convert from the selected species map to an ITIS taxonomic serial number.

CallType – The field will be interpreted as a call type. When the data in the field contains a forward slash “/”, anything after the / will be interpreted as a call subtype and will automatically be mapped to a `<Subtype>` element of `<Parameters>` if the user specifies a [*Parameters] section as described in section 3.2.8.

4.2.5 Conditional Entries

There may be times when certain XML elements should be created only when certain conditions are true in the source data. Consider a spreadsheet containing several rows of Deployment information, which will be translated into several Deployment documents for Tethys. Some of these deployments have had two distinct quality assurance tests run on them. These are represented by the columns QAStart_1, QAEnd_1, QACategory_1, QAStart_2, and so on, in the spreadsheet. These deployments would require two Entries for the `<Quality>` element.

However, most of the deployments were only quality tested once and their QAStart_2, QAEnd_2, etc, cells are empty. Creating empty `<Quality>` elements for these would result in an error, as `<Quality>` has mandatory children. The source data for such a case might look like:

QAStart_1	QAEnd_1	QACat_1	QAStart_2	QAEnd_2	QACat_2
7/18/2015 19:22	10/22/2015 16:18				
7/18/2015 19:46	10/21/2015 16:29	good	10/21/2015 16:30	10/22/2015 16:29	unusable
7/18/2015 19:56	10/21/2015 16:30	good	10/21/2015 16:31	10/22/2015 16:30	unusable
7/18/2015 20:03	10/22/2015 16:30				
7/18/2015 10:41	10/22/2015 16:31				

Figure 17 –Example deployment source document where quality assurance tests were run a second time on some deployments

Conditional entries allow us to create the elements *only when certain conditions are true*. This is done within the source map with a `<Condition>` element. `<Condition>` should encapsulate the element of interest and contain a `<Predicate>` which specifies the condition. Using the example we have above, an excerpt from the source map might look like:

```
<Condition>
      <Predicate>
              <Operand>[QAStart_2]</Operand>
              <Operation op="empty" retain="iffalse"/>
       </Predicate>
       <Quality>
              <Entry>
                     <Source> [QAStart_2] </Source>
                     <Kind> DateTime </Kind>
                     <Dest> Start </Dest>
              </Entry>
              <Entry>
                     <Source> [QAEnd_2] </Source>
                     <Kind> DateTime </Kind>
                     <Dest> End </Dest>
              </Entry>
              <Entry>
                     <Source> [QACat_2] </Source>
                     <Dest> Category </Dest>
                     <Default>unverified</Default>
              </Entry>
       </Quality>
</Condition>
```

We specify the column name to check in the **`<Operand>`** element. The condition we are checking is specified by the `op` attribute of the **`<Operation>`** element. Valid operations are:

- **empty** - checks if cell contents are empty. For numerical fields, 0 is considered to be empty.
- **matches** - checks if the cell contents match the value of the `<Operation>` element. For example, to only make note of compromised data:

```
<Predicate>
	<Operand>[QACat_1]</Operand>
	<Operation op="matches" retain="iftrue">compromised</Operation>
</Predicate>
```

- **matchesregexp** - checks the contents for the regular expression contained in the value of `<Operation>`. Regular expressions must follow Python's re module syntax (https://docs.python.org/2/library/re.html). For example, to import only deployments with projects that begin with TET

```
<Predicate>
	<Operand>[Project]</Operand>
	<Operation op="matchesregexp" retain="iftrue">^TET.*</Operation>
</Predicate>
```

Attribute **`retain`** is used to specify whether to include the element if the condition was true or false, designated by `"iftrue"` or `"iffalse"`.

4.2.6 Specifying attributes

Source maps can specify attributes of elements as well. As an example, for effort that is binned, it is necessary to specify an attribute indicating the bin size in minutes. An `<Entry>` can have additional fields marked `<Attribute>` that permit the specification of an attribute. Each `<Attribute>` supports the same `<Source>`, `<Default>`, and `<Dest>` tags as entries. The following provides an example that looks for the granularity in a source field called **Granularity** and then populates the **BinSize_min** attribute if it exists:

```
<Entry>
	<Source>[Granularity]</Source>
	<Default>binned</Default>
	<Dest>Granularity</Dest>
	<Attribute>
		<Source>[BinSize_min]</Source>
		<Dest>BinSize_min</Dest>
	</Attribute>
</Entry>
```

Optionally, we could have provided a **Default**, but if this source map were used for anything that was not binned data, the **BinSize_min** attribute would be set to the default value which would be inappropriate as the other granularity specifications (encounter, call) are not binned.

4.2.7 Accessing the name of the document

Some of the schemata require unique identifiers, usually in element `<Id>`. When uploading documents, we assign a special name, DefaultDocumentIdentifier, that takes the value of the filename being submitted without its extension. The name can be used in any `<Entry>` and is suitable for use in **Id** elements provided that the filename is unique. Example:

```
<Entry>
	<Source>[DefaultDocumentIdentifier]</Source>
	<Dest>Id</Dest>
```

```
</Entry>
```

Note that this is not an effective solution for when we retrieve information from a database. In that case, we are likely to be creating multiple documents. In such cases, values from the database queries can nearly always be used to form a unique Idenfitier.

4.2.8 Handling parameters

Data sources that have optional parameters are difficult to represent in tabular format, and we recommend generating XML directly using Nilus or other means when feasible. However, there are times when importing from a non-XML source is required and Tethys provides a limited method to handle data-specific parameters, although its use is discouraged. We currently assume that parameters are defined on a per species/call type basis, permitting one to measure call-specific parameters. For example, below we have an effort sheet where, for each species and call, there is a list of parameters:

Common Name	Species Code	Call	Parameter 1	Parameter 2
Sei Whale	Bb	Downsweep	Start_Hz	End_Hz
Sei Whale	Bb	LF Downsweep	Start_Hz	End_Hz
Sei Whale	Bb	All		
Sei Whale	Bb	Unspecified	Start_Hz	End_Hz
…				
Killer Whale	Oo	HFM	High_Hz	Low_Hz

The case-sensitive parameter columns are numbered and indicate that for Sei Whale downsweeps, the starting frequency will be the first parameter to be recorded. Similarly, for killer whales, the maximum frequency in Hz will be recorded as parameter 1. This definition of parameter positions is created by specifying the `<Table>` directive and adding the attribute `parameters="define"`. Within the Table, an entry with a `<Source>` of **[*Parameters]** and a `<Dest>` of **Parameters** is required as in this excerpt from SIO.SWAL.Detections.v1 source map from the sample database.

```
<Table query="select * from Effort$;" parameters="define">
<Effort>
   <Entry>
        <Source>[*Parameters]</Source>
        <Dest>Parameters</Dest>
   </Entry>
</Effort>
</Table>
```

To associate data with these parameters, values are placed in a subsequent Table that has columns named Parameter 1, Parameter 2, etc. and contains an entry mapping **[*Parameters]** to **Parameters**. For the above data, suppose one had a workbook with a sheet called Detections. If the source map had the following entry:

```
<Table query="select * from Detections$;">
<Detections>
   …<Entry>
        <Source>[*Parameters]</Source>
        <Dest>Parameters</Dest>
   </Entry> …
</Detections>
```

```
</Table>
```

with the following entries on the Detections sheet (start/end times and other fields not shown):

Species Code	Call	Parameter 1	Parameter 2
Oo	HFM	20000	38000
Oo	HFM	22000	39000

the values in the Parameter 1 column (20000 and 22000 shown here) would be associated with the element High_Hz of each entry for all killer whale HFM calls. Similarly, if a sei whale downsweep detection was present, any value in the Parameter 1 column would be associated with Start_Hz.

4.2.9 Multiple Sources and Nested Queries

It is possible to add data to Tethys from multiple sources. Suppose that a set of towed array deployments is maintained by passive acoustic monitoring staff and a separate ship-wide database contains ship position information. To construct a deployment document that contains information about the deployment such as sampling regime as well as trackline information, these two sources must be accessed together. Furthermore, if the trackline was for the entire voyage, one would need to begin processing the deployment and then query the portion of the trackline relevant to the deployment.

As with the other import methods specified above, source maps are used to provide direction on how data are to be translated. These use <Table> syntax, where data are queried using “select” commands (these are sequential query language (SQL) queries and people with knowledge of SQL may write more sophisticated queries). When the data are submitted, each data source is given a name that will be used to reference the data source from within the source map.