

WalesHER

GAT User Manual

Automated Data Upload (User Levels 0 & 1)

This document provides guidance on uploading datasets in csv format to WalesHER using the Load Data tool and migration SQLs in phpMyAdmin 3.3.0.

This guidance is for HER managers and other high level HER users only (user level 0-1) as it requires access to phpMyAdmin.

1. Preparing Datasets for Upload
 - 1.1 Converting to csv Format
 - 1.2 Column Names
2. Using the Load Data Tool
3. Editing Uploaded Datasets
4. Running Migration SQLs in phpMyAdmin 3.3.0
 - 4.1 Backing up Tables
 - 4.2 Migrating Data into her_core
 - 4.3 Migrating Data into her_core_event_link
 - 4.4 Migrating Data into her_core_site_type
 - 4.5 Migrating Data into her_source1_link
 - 4.6 Migrating Data into her_status
 - 4.7 Migrating Data into geo_locations
 - 4.8 Migrating Data into her_management_record
 - 4.9 Migrating Data into her_condition

1. Preparing Datasets for Upload

The Load Data tool will recognise data in the following formats: FoxPro, csv, xml, and dbf. This guide recommends using csv format as it is the only format that will not truncate large memo fields, which can affect the her_desc1 field in particular.

1.1 Converting to csv Format

Datasets deposited with the HER in xls or mdb format (Microsoft Excel/Access) will need to be converted to csv format.

To convert an xls file to csv double click on the file to open in Excel, go to 'File' and 'Save As' and chose 'CSV (MS-DOS)' from the 'Save as type' drop down list and click 'Save'.

To convert a mdb file to csv you will need to export the data to Excel, and then follow the steps as above. To export data from an Access database (2010) to Excel double click on the file to open in Access, click on the table you want to export, go to 'External Data' and chose 'Excel'.

1.2 Column Names

The column names in the csv file **do not** need to match the column names used in the WalesHER database. The migration SQLs (see section 4 below) will do this work for you. However spaces in column names should be replaced with an underscore e.g. *NGR qualifier* should be changed to *NGR_qualifier*. This can also be done once the csv file has been uploaded to WalesHER (see section 3 below).

2. Using the Load Data Tool

Login to WalesHER, and navigate to the 'Load Data' menu in the main panel. Click on 'CSV' and the Load Data tool will appear as shown below:

The screenshot shows the 'Load Data' tool interface. The left sidebar contains a menu with 'Load Data' selected. The main panel shows the 'Load: CSV' dialog box. Annotations with arrows point to specific elements:

- Use the browse tool to find and upload the data:** Points to the 'Select Upload File (max size:900M)' section where 'Corrugated_test.csv' is selected.
- Chose the database you want to upload the data to:** Points to the 'Database' dropdown menu showing 'herwales_gat'.
- Chose 'create permanent table':** Points to the 'Create or Add to Table' dropdown menu showing 'create permanent table'.
- Enter the table name. NB: - prefixing with 'aa' will place the table at the top of the list in the database, making it easy to find. NB: - Always use underscores, not spaces.** Points to the 'Table Name' field containing 'aa_corrugated_temp'.
- Leave all other options to their default settings:** Points to the 'Load Data Date Format' dropdown set to 'Auto' and other options like 'First Line As Column Names', 'Delimiting Character', 'Enclosing Character', 'Convert Datetimes to Date', and 'Perform Auto Inserts'.

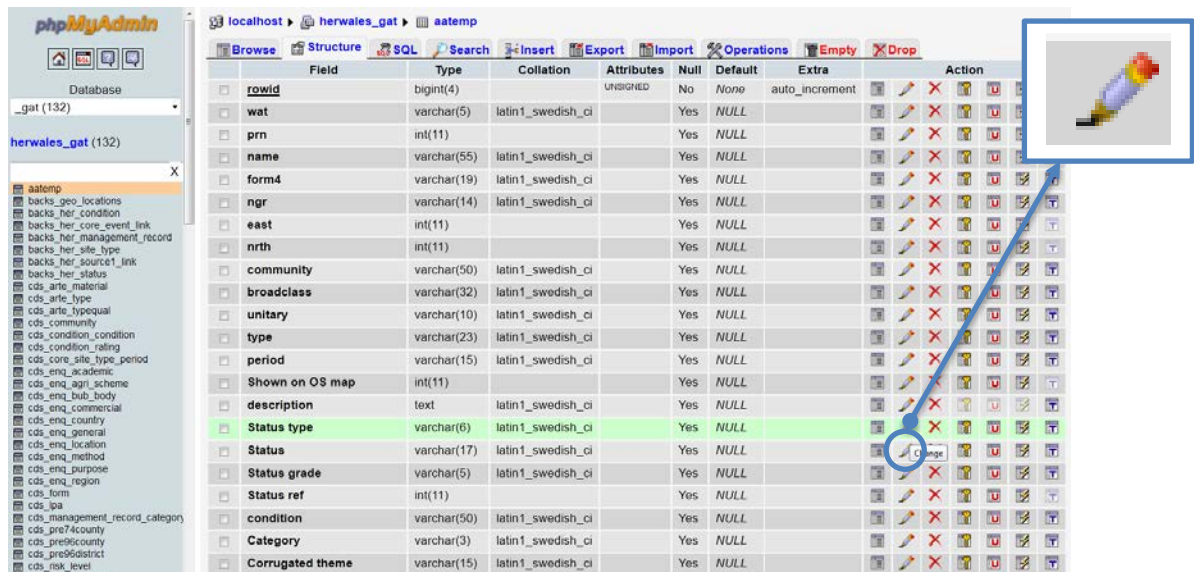
Click on the green tick to upload the file. Navigate to the 'Administrator' menu in the main panel and click on 'Database Admin 3.3.0' to open phpMyAdmin. In phpMyAdmin click on the database that you uploaded the csv file to from the list of database on the left hand side of the screen. Next check that the data has been uploaded correctly by finding the table in the list (if the table name is prefixed by 'aa' then it will appear at the top of the list), clicking on it and browsing the table as shown below:

The screenshot shows the phpMyAdmin interface. On the left, the 'Database' dropdown is set to 'herwales_gat (132)'. Below it, a list of tables is shown, including 'aatemps' and various 'backs' and 'cds' tables. The 'Table' dropdown is set to 'aatemps'. The main panel shows the 'Table' tab selected, displaying the table structure and content. A box labeled 'Table Contents' points to the table content area.

rowid	wat	prn	name	form4	ngr	east	nrth	community	broadclass
1	GAT	33317	Boat Store, Abersoch	Building - Roofed	SH3166528110	231665	328110		Maritime
2	GAT	33318	Garage, Abersoch	Building - Roofed	SH3137028150	231370	328150		Commercial
3	GAT	33319	Garages, Abersoch	Building - Roofed	SH3137028150	231370	328150		Transport
4	GAT	33320	Agricultural Buildings, Tal y Cafn	Building - Roofed	SH7881571720	278815	371720		Agriculture and Subsistence
5	GAT	33321	Agricultural Buildings, South of	Building - Roofed	SH8084569000	280845	369000		Agriculture and Subsistence

3. Editing Uploaded Datasets

It is possible to edit the structure of a new table uploaded to WalesHER in phpMyAdmin. For example you may want to change column names to remove odd characters or spaces between column names. To do this click on the table you have uploaded, go to the 'Structure' tab and click on the 'Change' tool on the column that you want to edit, as shown below:



This will take you to the edit page as shown below:

The screenshot shows the 'Edit Field' page for the 'Status ref' field in the 'aatemp' table. The field is currently an INT with a length of 11 and a default value of NULL. The 'Null' checkbox is checked, and the 'AUTO_INCREMENT' checkbox is unchecked.

Field	Status ref
Type	INT
Length/Values ¹	11
Default ²	NULL
Collation	
Attributes	
Null	<input checked="" type="checkbox"/>
AUTO_INCREMENT	<input type="checkbox"/>
Comments	
MIME type	
Browser transformation	
Transformation options ³	

Edit the fields as appropriate, and click 'Save' to save your changes. To abandon any edits simply click the back button in your browser, or navigate to another table or database.

4. Running Migration SQLs in phpMyAdmin 3.3.0

The migration SQLs are a pre-prepared set of SQLs designed to move datasets from within one table to another in WalesHER quickly and efficiently. The SQLs are designed to be altered and tailored to suit different datasets by the HER manager. Use the SQLs below as a basis to create tailored SQLs to fit the datasets that you want to upload with the tables in WalesHER.

NB Take care to ensure that the 'restricted' field is populated appropriately: 1 = restricted, 0 = unrestricted. ***All core finds records and all metal detecting event records must be flagged as restricted***

NB The following columns should be populated with the following values where they occur in every table:

wat	GAT
compiler	Name of the person entering the record to the HER e.g. Angharad Stockwell
compiler	GAT
origin	GAT
copyright	GAT

NB The 'compiledon' column in all tables must always show the date on which the record is added to the HER. This is auto-filled when adding records manually in the panel view. The 'now()' clause in the SQLs shown below will also do this automatically.

4.1 Backing up Tables

Once you have uploaded your dataset to WalesHER using the steps above, you are ready to migrate the data it contains to the relevant tables in the WalesHER databases.

***It is very important to back up the tables you want to migrate data into before running the migration SQLs*.** This will ensure that errors can be quickly rectified by restoring backup tables. To back up tables in WalesHER follow the steps for copying tables provided in the *WalesHER Moving & Copying Tables* document.

NB Prefixing the backup table names with 'backup' will ensure that they appear together in the list of tables in phpMyAdmin, making them easier to find when restoring or deleting them later on e.g. *backup_her_core*

NB It is a good idea to keep the backup tables for at least a week after completing any data migration to allow for migration errors to be identified.

To restore a backup table simply delete the table that has been amended, and rename the backup table e.g. *backup_her_core* becomes *her_core*.

4.2 Migrating Data into her_core

The SQL below will insert data from selected columns in a table to her_core:

```
INSERT INTO her_core
(wat,prn,watprn,name,desc_1,compiler,compilero,compiledon,origin,copyright,restricted)

SELECT 'GAT',prn,concat('GAT',prn),name,concat(description,' (Batten, 2011)'),'Angharad
Stockwell','GAT',now(),'GAT','GAT',0

FROM aa_corrugated_temp
```

The first line of the SQL points to the table and table columns in her_core that you want to insert data *into* (the column names must match those in her_core). The second line of the SQL points to the table and table columns that you want to draw the data *from*.

The order in which the column names appear in the first and second lines of the SQL must match. For example in the SQL above the column 'wat' in the her_core table will be populated with the

values contained in the column 'GAT' from the aa_corrugated_temp table because the column 'wat' is looking to the column 'GAT' for the data.

Where the value will be the same for each record in a column it is possible to specify that value in the SQL. This applies to all instances in line two where text values are enclosed in apostrophes ("), or a number has been specified (numbers do not need apostrophes). For example in the SQL above the 'compiler' column in her_core will always be populated with the value 'Angharad Stockwell', and the 'restricted' column will always be populated with the value 0 in each record.

The following section in line two of the SQL above is asking WalesHER to insert (Batten, 2011) after the last line of text in the desc_1 field in order to link the description to its source using the Harvard referencing system: *concat(description, ' (Batten, 2011)')*

After you have backed up her_core, amend the SQL as appropriate to suit your dataset and run in phpMyAdmin. New data from your uploaded table should now appear in her_core.

4.3 Migrating Data into her_core_event_link

The SQL below will insert data from selected columns in a table to her_core_event_link:

```
INSERT INTO her_core_event_link (wat,watprn,ewatprn,compiler,compiler0,compiledon,origin,copyright)
SELECT 'GAT',concat('GAT',prn),concat('GAT',44367),'Angharad Stockwell','GAT',now(),'GAT','GAT'
FROM aa_corrugated_temp
```

This SQL will link an event record in the her_event table to multiple core PRNs. To do this the event record must already be present in her_event. Make a note of the event PRN that you want to link to your dataset.

Enter the event PRN to the second line of the SQL as shown above. Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data.

After you have backed up her_core_event_link and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in her_core_event_link.

4.4 Migrating Data into her_core_site_type

The SQL below will insert data from selected columns in a table to her_core_site_type:

```
INSERT INTO her_core_site_type
(wat,prn,watprn,broadclass,type,period,compiler,compiler0,compiledon,origin,copyright,eventprn,rank)
SELECT 'GAT',prn,concat('GAT',prn),broadclass,type,period,'Angharad
Stockwell','GAT',now(),'GAT','GAT',44367,1
FROM aa_corrugated_temp
```

Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data.

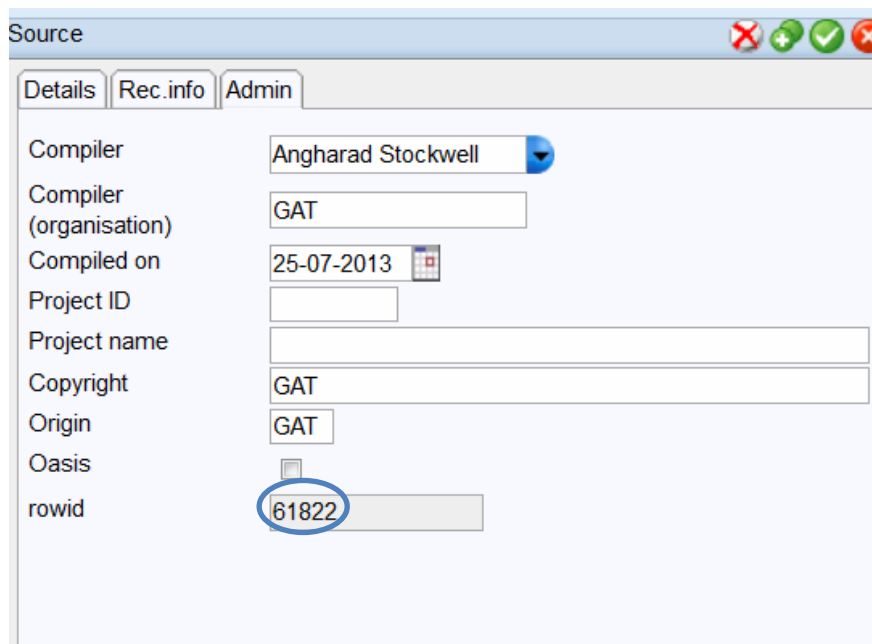
After you have backed up her_core_site_type and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in her_core_site_type.

4.5 Migrating Data into her_source1_link

The SQL below will insert data from selected columns in a table to her_source1_link:

```
INSERT INTO her_source1_link (wat,prn,watprn,source_id,compiler,compilero,compiledon,origin,copyright)
SELECT 'GAT',prn,concat('GAT',prn),61822,'Angharad Stockwell','GAT',now(),'GAT','GAT'
FROM aa_corrugated_temp
```

This SQL will link a bibliographic record in the her_source1 table to multiple PRNs. To do this the bibliographic record must already be present in her_source1. Make a note of the bibliographic record's row ID, this can be found in the 'rowid' column in the her_source1 table in phpMyAdmin, or in the 'rowid' field in the 'Admin' tab in the panel view, as shown below:



The screenshot shows the 'Source' table in phpMyAdmin, specifically the 'Admin' tab. The table has several fields: Compiler (Angharad Stockwell), Compiler (organisation) (GAT), Compiled on (25-07-2013), Project ID, Project name, Copyright (GAT), Origin (GAT), Oasis, and rowid (61822). The 'rowid' field is circled in blue, indicating its importance for the SQL query.

Enter the source's row ID to the second line of the SQL to correspond with 'source_id' in the first line. Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data.

After you have backed up her_source1_link and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in her_source1_link.

4.6 Migrating Data into her_status

The SQL below will insert data from selected columns in a table to her_status:

```

INSERT INTO her_status
(wat,prn,watprn,status,status_typ,grade,reference,compiler,compilero,compiledon,origin,copyright)

SELECT 'GAT',prn,concat('GAT',prn),status,`status type`,`status grade`,`status ref`,`Angharad
Stockwell`,`GAT',now(),`GAT`,`GAT`

FROM aa_corrugated_temp

WHERE length(trim(`status type`))>0

```

The WHERE clause in this SQL is looking for values greater than 0, therefore the status table will only be populated for records with status information attached to them.

Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data.

After you have backed up her_status and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in her_status.

4.7 Migrating Data into geo_locations

The SQL below will insert data from selected columns in a table to geo_locations:

```

INSERT INTO geo_locations
(geo,`get`,wat,prn,watprn,ngr,east,nrth,community,unitary,compiler,compilero,compiledon,origin,copyright,rec_type)

SELECT
geomFromWKB(point(east,nrth)),0,'GAT',prn,concat('GAT',prn),ngr,east,nrth,community,unitary,'Angharad
Stockwell`,`GAT',now(),`GAT`,`GAT`,`CORE`

FROM aa_corrugated_temp

WHERE east>100000 and nrth>100000

```

The WHERE clause in this SQL is looking for eastings and northings that have values greater than 100000 in order to ensure that the locations data is a valid.

Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data.

After you have backed up geo_locations and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in geo_locations.

If essential geo_locations data is missing from your uploaded dataset such as then simply run a geo spatial query to populate the blank columns. Please refer to the *WalesHER GIS Mapper* document for instructions on how to do this. The following geo_locations columns can currently be auto-populated in this way; community, postcode, lpa, and unitary.

4.8 Migrating Data into her_management_record

The SQL below will insert data from selected columns in a table to her_management_record:


```

INSERT INTO her_management_record
(wat,prn,watprn,category,rec_by,rec_year,compiler,compilero,compiledon,origin,copyright,eventprn)

SELECT 'GAT',prn,concat('GAT',prn),category,'Batten, A.', '2011','Angharad
Stockwell','GAT',now(),'GAT','GAT',44367

FROM aa_corrugated_temp

```

Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data. After you have backed up her_management_record and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in her_management_record.

4.9 Migrating Data into her_condition

The SQL below will insert data from selected columns in a table to her_condition:

```

INSERT INTO her_condition
(wat,prn,watprn,record_by,year,condesc,compiler,compilero,compiledon,origin,copyright,eventprn)

SELECT 'GAT',prn,concat('GAT',prn),'Batten, A.', '2011',management_observation,'Angharad
Stockwell','GAT',now(),'GAT','GAT',44367

FROM aa_corrugated_temp

```

Ensure that the SQL is looking to the correct columns in the uploaded dataset for its data.

After you have backed up her_condition and amended the SQL as appropriate to suit your dataset, run the SQL in phpMyAdmin. New data from your uploaded table should now appear in her_condition.