WHONET Automation Tool User Guide

WHO Collaborating Centre for Surveillance of Antimicrobial Resistance

Boston, November 2023

Table of Contents

About this document 3

Configuration overview 3

Understanding the date-stamp syntax 3

Date-stamp patterns 4

Relative date modifiers 4

Automatic file name matching 5

Automation Tool step configuration details 5

1. Copy input 5

2. Pre-parser 6

3. BacLink 6

4. Monitor row count 7

5A. Split output 8

5B. Concatenate output 8

6. Copy intermediate file 8

7. WHONET 8

8. Copy output 9

9. Concatenate extracts 9

10. Backup 9

11. Email results 9

12. Monitoring alerts 10

E-mail configuration 11

Code mapping assistant 12

Automation monitoring 14

Monitor settings 14

Automation monitor 15

View automation log 15

Manual data processing 15

Other ways to invoke the Automation Tool 16

About this document

This guide provides information about how to configure the WHONET Automation Tool. For a description of the tool and help with installation, please see the installation guide found at this link: https://whonet.org/WebDocs/WHONET_Automation_Tool_Installation_Guide.pdf

Configuration overview

This system breaks the overall data processing pipeline into many different discrete steps. Most users will only need to configure and enable a subset of steps which correspond to their specific needs. The steps are executed in numeric sequence, and only those steps which are enabled are executed. This guide covers the configuration of the full sequence of steps, as well as information about the other functions of the utility such as optional email configuration, process monitoring, and manual data processing.

The tool is designed to replace the repetitive manual steps needed to process an automated data extract on a regular basis with an automated process that accomplishes the same goals. For example, if your hospital has already setup a microbiology data extract from an LIS system, often with the current date in the file name, this file must be processed with BacLink and potentially aggregated into a monthly or yearly data file for easier use. The tool is designed to handle this scenario, and this document will proceed under the assumption that you already have this automated data feed extract, as well as the associated BacLink and WHONET configurations needed to do the processing manually. These are prerequisites for the automation process and are covered in separate guides found in the training section of our website: https://whonet.org/training.html

The tool is designed to be as flexible as possible, and as such there is a necessary level of complexity required to accommodate the various situations you might encounter in practice. The primary concern is the issue of variable dates in the file name of input and output files for various steps of the process. For example, at our hospital we receive a date-stamped file every day automatically in the same folder. We want to process this single day of data with BacLink to standardize the codes and convert it into the WHONET format. We also wish to aggregate the data into a yearly file since hundreds of daily files quickly becomes unwieldy. The rest of this document will use the common scenario mentioned here as a basis for providing examples of the corresponding syntax recognized by the tool. Your settings will vary according to the details of your data feed and needs. If your situation diverges from this by too much, then the Automation Tool may not meet your needs. Please write to us at help@whonet.org or create a post on our community website at https://community.whonet.org to suggest new features that might help us improve the system.

Understanding the date-stamp syntax

The most challenging aspect of Automation Tool configuration is the date-stamp syntax required to capture the variable part of the file names for input and output files. We have designed the tool to be very flexible so that it can be used with a wide variety of date formats. The basic principle is that a file name might contain static text as well as a variable part which corresponds to the date in some fixed format. For example, your daily microbiology extract file might be named “Micro-Data-2023-12-31.txt”. The following morning, this file name would adjust to “Micro-Data-2024-01-01.txt” when the new file is deposited. The automation tool needs to know which file to process on a given schedule invocation, so

the date syntax described here allows you to provide the expected file name patterns in a way that continues to work as time progresses.

For the example above, we would use this pattern inside the tool's arguments "<Micro-Data-{yyyy}-{MM}-{dd}.txt>". The angle brackets (<, >) let the system know that the contained text has a date pattern which requires deciphering and substitution. When the system "wakes up" according to the given schedule and begins processing, it will note the current date according to the computer's system time. It will then translate this date into the corresponding format as specified by the argument template shown above contained in the angle brackets. The angle brackets themselves will be removed from the actual command text that is sent to the underlying application. For example BacLink would receive "Micro-Data-2024-01-01.txt" on that date.

Date-stamp patterns

There are multiple ways to represent dates which are reflected with the following patterns. All of these identifiers are enclosed in curly braces ({, }) to distinguish them from static text. These curly braces will also be removed from the command text.

1. {yyyy} – Four-digit year.
2. {yy} – Two-digit year.
3. {MM} – Two-digit month.
4. {MMM} – Three-letter abbreviation for the month ("Jan", "Feb", etc.)
5. {dd} – Two-digit day of the month. Ex. 05, 12
6. {d} – One or Two-digit day of the month. Ex. 5, 12

These patterns are combined with static text to match the file name patterns you need. For example, if your input files are named similarly to this: "24_01_13.txt", then the corresponding pattern would be as follows: "<{yy}_{MM}_{dd}.txt>"

Relative date modifiers

Sometimes the date stamp for a file name will not match the actual creation date of the data. For example, if your files are deposited every morning at 4 AM with the prior day's date in the file name instead of the current date. In this situation a relative date must be used with respect to the "run date" of the process. For example, on the morning of 2024/01/01, a new file is deposited on the network drive with a date corresponding to the data contained, rather than the date of deposition, namely 2023/12/31.

More concretely, the file name will be "2023-12-31.txt" on the morning of 2024/01/01 instead of "2024-01-01.txt". When processing begins later on 2024/01/01, we need to adjust the date by subtracting 1 day from the system run date and substituting that calculated date in the argument template instead of the current run date. To accomplish this, we provide a simple modifier in the "day of the month" part of the argument template as shown: "<{yyyy}-{MM}-{dd-1}.txt>"

Relative date modifier patterns are as follows, where "#" must be replaced with an integer value (1, 2, 3, etc.)

1. {dd-#} – Subtract "#" days from the system run date.
2. {d-#} – Subtract "#" days from the system run date, but only use the "one-digit" day of the month when applicable (for dates prior to the 10th of each month).

Automatic file name matching

The above two sections take care of most scenarios you might see in practice with the important exception of time values being incorporated into the file name. Because the precise time is not important to us, and will vary from day to day, we do not provide patterns to attempt to capture this. We would not know the correct value to substitute there since it is highly variable in practice. For example, if today's file is named "2024-05-13_00.04.32.352.txt" we recognize that the time portion of this will vary considerably from day to day and is also irrelevant for our data processing purposes. Sometimes the fractional component of seconds is included in the file name as shown.

To resolve this problem we include the "*" match operator, which works similarly to how you may be familiar with it when searching in the Windows file browser. For example, searching for "Presentation*.docx" will match all files beginning with "Presentation" and ending with ".docx".

For our purposes, you can use the "*" operator to match input files that contain a time stamp in addition to a known date as in the ".txt" example provided. The automation tool argument template which will match "2024-05-13_00.04.32.352.txt" would be "<{yyyy}-{MM}-{dd}_*.txt>"

This pattern would also match a file deposited the following morning at a slightly different time, such as: "2024-05-14_00.04.28.582.txt"

Please note that if multiple files match this pattern for a given concrete date (two files from May 13th with different times), the Automation Tool will sort them alphabetically and take the last one. This reflects the assumption that if two exports were generated on the same day but at different times, we probably want the last one (a correction to the first file).

Automation Tool step configuration details

You must only enable and configure the steps which correspond to your needs. The enabled steps will be executed in numeric sequence. If a step fails, the subsequent steps will not be executed on that day and an error message will optionally be sent reflecting the encountered issue. The information below provides details on how to configure each step.

Important note: You MUST press the "Save settings" button to store the changes you've made into the system before closing the interface or launching a manual run using the "Execute date range" button on the interface. Only the stored settings will be used for processing.

1. Copy input

Purpose: To copy input files from a network drive to the local hard drive. This step is not required, but attempting to process files over a network directly can cause random errors due to network connectivity issues.

"Input file template" should refer to only a single file, likely utilizing the date templates discussed above.

Ex. <"\\GRT.Hosp.org\Microbiology\Data export\DailyData-{yyyy}(MM}{dd}.csv">

"Output file template" should refer to a location and file name where you would like the above file to be copied for processing. Suppose you've created a folder at "C:\WHONET\Input\" where you would like to

temporarily copy the network file mentioned in the input to. In this case, the “output file template” would be as follows, noting that the file is renamed in this step for simplicity of the remaining steps.

Ex. <”C:\WHONET\Input\{yyyy}{MM}{dd}.csv”>

The result is that the system will copy a file matching the input file template to the output file template folder and naming convention specified.

“Copy additional file” – If this sub step is enabled, a second file can be copied. For example, if your process has two daily input files from two different sources. The input and output file templates behave the same as described above.

2. Pre-parser

Purpose: The “pre-parser” step is used to configure any purpose-built applications you might have that would make the raw input file able to be processed by BacLink. For example, if your data file cannot be processed by BacLink without running some process on the raw data to convert its structure, then this step would allow you to configure that. This step is not commonly needed, and if you can process your input data directly with BacLink, then you may simply leave it disabled.

“Parser” – You must use the “Browse” button to the right of the text box to select an executable file which will do the pre-parsing job you require.

“Arg template” – This is the command line argument template for your pre-parser executable. Typically it would include both an input and output file in the argument template according to the manufacturer’s specifications and using the date template patterns already described.

“Delete pre-parser input file” – when enabled, this would allow you to delete a temporary file named in the adjacent text box if the pre-parser executable runs successfully. Note that you should never delete the original source data files, for example, those on the network drive. Only consider deleting copies you’ve made in other temporary locations.

“Delete pre-parser ADT file” – This would allow you to delete a second temporary file. For example, our pre-parser at our hospital merges the separate microbiology and admission/discharge/transfer data (which come from different, incompatible systems) into a single file. If that merge process (the pre-parser) is successful, we want to clean up both temporary files we have copied to our local disk in “1. Copy input”.

“Additional parser” – This sub step allows one to configure another “pre-parser” if this is required.

3. BacLink

Purpose: To run BacLink, which will convert your raw input data into a WHONET file with standardized codes using BacLink’s command-line interface.

BacLink’s command-line interface is as follows:

BacLink_Config_File Input_File Output_File

BacLink must be configured separately and should be working well prior to any attempt at automation. Please see our documentation on this subject at the training info link provided near the top of this document.

“BacLink” – Use the “Browse” button to select BacLink.exe, which is typically located at C:\WHONET\BacLink.exe by default.

“Arg template” – The adjacent text box allows you to specify the config file, input, and output files for BacLink to run in “batch” or “automated” mode. This means that when properly configured, BacLink can process the input into the output according to the details provided by the configuration without user interaction. An example of the argument template is provided here:

"C:\WHONET\BWH-Sunquest.cfg" <"C:\WHONET\Input\B{yyyy}{MM}{dd}.txt">
<"C:\WHONET\Data\{yyyy}{MM}{dd}.sqlite">

This sample argument template says that BacLink should use the config file “BWH-Sunquest.cfg” to process the input file (ending with .txt) into the WHONET output file (ending with .sqlite).

“Delete input file” – If BacLink is successful in its conversion, you may wish to delete the input file, for example if it was a temporary file copied locally through “1. Copy input”. Note that you should never delete the original source data files. Only consider deleting temporary copies if applicable.

“Enable 2nd BacLink run” – This allows you to run a second distinct BacLink process using a different argument template and configuration file if this is necessary for you.

4. Monitor row count

Purpose: The Automation Tool has the capability of monitoring the daily isolate count for input files. This can be utilized by the monitoring alerts in step “12. Monitoring alerts” to send a designated list of recipients a message if the daily isolate count is too high or too low, which might indicate a problem that requires further investigation.

This section only describes the necessary configuration for this step. The overall automation monitoring system must be configured as described in the section “Automation monitoring” found below this document.

“BacLink output” – This argument template should refer to the output of the BacLink step. In the example above from “3. BacLink”, that would refer to the “.sqlite” WHONET file. This step will read this file, which only has the daily isolate count at this stage and record that value in a table as configured in the automation monitoring setup.

Ex. <"C:\WHONET\Data\{yyyy}{MM}{dd}.sqlite">

“Institution column name” – You can specify the WHONET column name (typically INSTITUT) you would like to use to stratify the isolate counts in the automation monitoring database. For example, if your data represents isolates from 10 different facilities (as stratified by INSTITUT), but only one of them has a low row count, this will allow you to capture that information and alert on this condition.

“Process input file” – This should name a raw process input file you would like to monitor, for example the BacLink input file. The file named here will have its cryptographic hash computed and stored in the automation monitoring database. The reason for this is so that we can quickly and easily detect identical duplicates of input files which might not have the same name. For example, if the export system has a problem and produces an identical copy of the data from yesterday, the cryptographic hash will match, which is an alert condition in “12. Monitoring alerts”.

5A. Split output

Purpose: This step allows you to break up the input file using the “Institution column”, or another stratifying column name you specify, into may files. This step is not commonly needed. If you believe you need this step, please contact us via email for assistance with generating the crosswalk file.

5B. Concatenate output

Purpose: To consolidate the daily BacLink output file into a monthly or yearly data file.

Typically in automated setups, the data processed by BacLink only represents results made available on a single day at the hospital. This step allows you to append these results to a monthly or yearly data file for more convenient usage.

“Input file template” – This refers to the daily BacLink output file from “3. BacLink”

Ex. <"C:\WHONET\Data\{yyyy}{MM}{dd}.sqlite">

“Output file template” – This refers to the name of the aggregated data file you would like to append to. If this file doesn’t exist it will be created, for example, when the year changes in the example below.

Ex. <"C:\WHONET\Data\BWH-{yyyy}.sqlite">

Note that the pattern above does **not** include the month or year portions of the date. This will have the effect of using the same output file name for the entire year, for example “BWH-2024.sqlite”. When the year changes, the system will automatically begin a new file for 2025.

“Concatenate second BacLink output to file” – This allows you to append or concatenate a second file somewhere, for example if you needed two BacLink processes in “3. BacLink”.

6. Copy intermediate file

Purpose: This step is intended to copy the output file of “3. BacLink” or “5B. Concatenate output” to a network share so that other users can see and work with an updated copy for their own purposes.

“Input file template” – Either the BacLink output file or the result of the concatenation step (yearly file).

Ex. <"C:\WHONET\Data\BWH-{yyyy}.sqlite">

“Output file template” – Location and file name where you would like the backup copy to be sent.

Ex. <"\\Your.Network.org\MicroData\WHONET_{yyyy}BWH.sqlite">

7. WHONET

Purpose: To run a WHONET report with the latest data as prepared by the previous steps. This step allows you to run WHONET in command-line or “batch” mode, which means that it will run without a user interface and perform the tasks specified by the report file you’ve created.

WHONET must be configured separately. Please see the WHONET documentation available at the training link provided at the top of this document.

WHONET’s command-line interface for user-defined report files is:

Automation_report_file.rpt TODAY=yyyy/MM/dd

The “TODAY” part of the line above is a special syntax used in conjunction with the Automation Tool. Some WHONET analyses need to know what the current date is (to calculate a relative date for SaTScan analyses to start and end for example). When the TODAY argument is omitted, the real system date will be used whenever necessary. If the TODAY argument is provided however, we can override that behavior. You would need that if you decided to manually process a range of dates using the Automation Tool interactively, for example. You only need to specify the run date for SaTScan reports.

“Delete input file” – This allows you to delete a certain file if WHONET is executes successfully. This may not be applicable for your use case.

8. Copy output

Purpose: Typically used to copy an output file from the WHONET step to a network location so that others can view it.

“Input file template” – Specifies a source file to be copied, such as the analysis output file from the WHONET report.

“Output file template” – Specifies the destination copy.

9. Concatenate extracts

Purpose: This step can aggregate many text-based outputs into a single feed. This step is likely not relevant for your needs and was designed with a specific project in mind.

10. Backup

Purpose: This step backs up an entire folder tree to some other specified location.

“Input path” – The folder to be copied.

“Output path” – The location to copy to.

“Enable secondary backup” – This allows you to specify a second folder tree to copy to another destination.

11. Email results

Purpose: To notify a specified list of email addresses that the automation process has completed, and optionally send them an encrypted file.

“Recipients” – A comma-delimited list of email addresses to send to.

“Subject” – The subject of the email to be sent.

“Email body” – The text of the email to be sent.

“Encrypt and attach file” – This is NOT suitable for sensitive patient data and is not recommended generally to prevent accidental data sharing. A better solution for most (and perhaps required by your applicable laws or hospital policies) is to copy the data to an internal network location accessible by your intended recipients using “8. Copy output” and to use this email feature to simply notify them that they can check the location for their data. In other words, you can notify users that there are results to view, including where to find them, but without including the results themselves as an attachment.

If you choose to setup the encrypted attachment in acknowledgement of these limitations and legal implications, you can specify the file encryption password to use as well as the “File template” which would likely name an output file from your analysis in “7. WHONET”.

The named file will be encrypted with the “7-zip” software using the password provided. Your users must also have this freely available software on their computer and know the password to decrypt it.

Note that the password must not be included in the email itself and should be shared directly with the recipients through other means.

12. Monitoring alerts

Purpose: This step is used in conjunction with the “automation monitor” database which must be configured prior to enabling this step. This step will look for several different alert conditions based on the run time of the individual steps, the isolate row count for today, as well as sending a daily summary of the processes even when there are no issues detected.

“Recipients” – A comma-delimited list of email addresses to send the reports to.

“Always send a daily process summary report” - The daily summary will include attachments showing how today’s run times and isolate counts compare with the historical background. If one of the steps resulted in an error, your recipient list will instead receive an email to that effect.

“Send an additional alert when unexpected runtimes or row-counts are detected” – This option allows you to specify the sensitivity for triggering alerts. For example, if today’s measurement (run time for an individual step or the isolate count for today) is more than one or two standard deviations away from the mean. We recommend using two standard deviations to reduce the number of alerts, which will dilute their effect.

“Suppress alerts associated with non-priority processing steps” – Suppress alerts for the following steps: 5A, 5B, 6, 8, 9. Errors with these steps are still reported, but an atypically high or low runtime will not be.

“Suppress high row count and high runtime warnings” – If you don’t consider high runtimes or isolate counts to be an issue you would like to be made aware of, you may suppress the associated emails. Low values are always alert conditions. For example, if a process that ordinarily takes 20 seconds to run completes in a fraction of a second on a given day, this would usually merit investigation.

“Notify recipients when an output file changes” – This will scan a particular file for changes from day to day by calculating the cryptographic hash for the file and comparing it against earlier hashes. If the file changes, the cryptographic hash will change, and the specified emails will be alerted. In this case we are expecting the output file to remain static from day to day, so we alert when it has changed. This works well when the monitored file represents something that doesn’t change often, such as analysis results for a cluster analysis (changes when a new signal is discovered), or for a “microbiological alerts” analysis used to detect priority species or resistance (changes when a new priority issue is discovered). It would not be useful for analyses with output that is expected to vary, however little, from day to day as you would always receive an alert in that case.

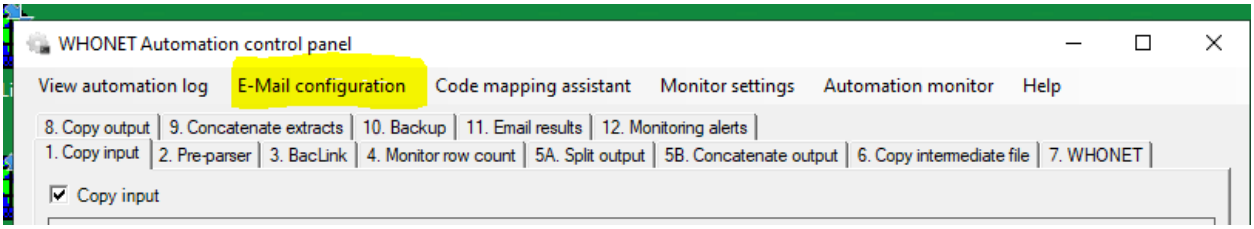
“File name” – The file to monitor for changes.

“Recipients” – The comma-delimited list of emails to notify if the monitored file changes.

E-mail configuration

If you would like to configure the Automation Tool to send email alerts to your users, including errors, warnings, and reports, you must have the associated credentials and server details for an SMTP email server. We use Amazon WorkMail for our service provider, but any SMTP server that is accessible by the system running the Automation Tool should work. The details of setting up an SMTP server are out of scope for this document. Your information technology department may be able to provide these details for you.

To begin configuring the Automation Tool for email, choose “E-Mail configuration” from the top bar of the interface as shown:



The settings shown on the Email configuration form must be fully-populated for the system to work.

“Enable email notifications” – This checkbox activates the email system and allows you to enter the SMTP server details.

“Notify on errors” – The adjacent text box will accept a comma-delimited list of email addresses that should be notified if there is a system error during automated processing. These users will be notified regardless of whether step “12. Monitoring alerts” has been configured.

“Notify on new codes” – When new BacLink codes are discovered, a maintainer must periodically define them and reprocess the data as described in the “Code mapping assistant” section.

“Sending email address” – This is the email account that will be used to send the email messages from the Automation Tool to your recipeients. We use a dedicated inbox for this purpose.

“SMTP host” – This is the DNS or IP address of the SMTP server.

Ex. email-smtp.us-east-1.amazonaws.com or 54.123.59.86

“SMTP port” – The port number that the SMTP server listens for connections on.

Ex. 25, 587, 2525.

“SMTP Username” – The SMTP username that should be used to connect to the mail server.

“SMTP Password” – The SMTP password associated with the above username.

“Use SSL encryption” – We recommend that this is always turned on, but it will depend on your use case and available options. This setting typically also affects the “SMTP port”.

Ex. Non-secure connections use 25 typically, SSL-encrypted connections might use 587.

“New code notify interval” – The minimum number of days to wait before notifying the designated recipients about new BacLink codes. For example, 7 days would mean that they should only be notified at most once per week.

“Save” – You must press Save whenever you make changes to the email settings or before attempting to send a test.

“Cancel” – Close the form without saving.

“Send test email” – This will attempt to send a sample email to the recipients listed above in both the “errors” and “new codes” lists. Please ask that the users listed check their inboxes. You may wish to use your own inbox only while trying to get the email settings correct. Once you can receive emails successfully from the test, you can then replace the two email address lists with their permanent values and save the settings.

Code mapping assistant

The code mapping assistant is an optional feature which tries to manage the backup copies for data files necessary to “fill in” empty spaces in the WHONET data files associated with unmapped codes. For example, if “sputumSpec” is not defined in BacLink as being matched with the WHONET specimen type “Sputum”, the WHONET data file will have a blank space for the specimen type here. When the BacLink administrator eventually defines this code, future data files from that point onward will have it properly populated. The code mapping assistant tries to provide a solution for the data in the period between when the “sputumSpec” code was first seen in the data and when it was defined in BacLink.

You may of course manage your data without using the code mapping assistant. You will need to follow similar steps to those outlined here, but you will need to do them manually, including rerunning the Automation Tool for the range of dates since the last code mapping session.

It does this by maintaining a copy of the production data along with the date (for example if you create a yearly data file using “5B. Concatenate output”). When the administrator periodically decides to map the newly discovered codes, the current production file is backed up and set aside. The previous production file backup is copied to the production location. After the user defines the codes in BacLink, the Automation Tool will run its procedures sequentially from the “last mapping date” associated with the production backup until the current date, but it will omit any analysis steps. In other words, after mapping the codes, the Automation Tool will reprocess the full date range again to regenerate the data.

The resulting data file should contain the newly defined codes. This new file then becomes the production backup associated with the current date and is stored in a folder for this purpose.

The code mapping assistant tries to make this complex process easier by guiding the administrator through a sequence of 5 steps which are described below.

Step 1. – Define the unrecognized codes using BacLink

This step provides a link to open the BacLink executable. It then instructs the administrator to proceed to define the unrecognized codes normally using the BacLink interface and return to the code mapping assistant when finished.

Step 2. – Create a production backup of the current data file in case there's a problem.

There are a few details that must be filled in here and saved.

"Data file folder" – This is the location that you ordinarily keep your main WHONET data file. If you are using "5B", this would be the same location for the "yearly" data file. Ex. C:\WHONET\Data\

"Data file template" – Similarly, this would typically be the data file template associated with the "yearly" file in "5B". Ex. <BWH-{yyyy}.sqlite>

"Backup folder" – This should be a newly created, empty folder for this purpose. Ex. C:\WHONET\Data\Archive\ProductionBackup\

Please remember to press "Save settings" near the bottom of the screen whenever you make changes.

The details above only need to be set once. After they are correctly set and saved, the only action the administrator needs to take for "2" is to press "Create production backup", which as it indicates will simply copy the file matching the data file template into the backup folder provided.

Step 3. – Overwrite production data files with snapshots

A snapshot is the completed data file for the last time the administrator used this utility to map codes. In other words, after defining the codes previously and processing the data to that date, you should have a fully populated WHONET data file that can serve as the basis for the next code mapping session's start point.

"Snapshot folder" – This should be set to a newly created, empty folder on initial setup. Ex. C:\WHONET\Data\Archive\Snapshot\

Please remember to press "Save settings" below to record your change.

After setting the snapshot folder above and saving the settings, or on any subsequent use of the code mapping assistant, the administrator only needs to press the "Overwrite" button near the bottom of this step. This will copy the last completed data file to the production location, in our example C:\WHONET\Data from Step 2 of the code mapping assistant sequence.

Step 4. – Reprocess data files

Now that the previous snapshot has been moved to the production location expected by the Automation Tool's main sequence of steps, we can re-run the data aggregation steps again over the date range that corresponds to the last time the code mapping assistant was used. This will regenerate and aggregate the data as defined by your configuration, but it will not re-run the analyses or generate emails to your users. The result should be that you have a complete production data file given that you've defined the unrecognized codes accumulated since the last code mapping session. This new file then becomes the new "snapshot" file for next time in the next step.

You will note that you have the option to set a "Previous code mapping session" date. If this is the first time you've used the code mapping assistant, you should set this date to match the first date you have input source data files. For example, if you started to receive raw data file extracts on June 7th of 2024, then that would be the appropriate date for this value. Put more simply, this is the "start date" which

the Automation Tool will use to run the data aggregation steps, and the current date will be used as the "end date".

After inputting the date for the first time (and saving it), or on any subsequent use of the assistant, all the administrator needs to do is to press "Reprocess data files". This will run the first (up to) 5B steps in sequence once for every date in the range from the "Previous code mapping session" to the current date.

If this entire process succeeds, you may move on to step 5. If not, you should press "Restore production backup" from the lower left of the screen which will copy the file you backed up in "2" of this sequence. You would then need to manually diagnose and correct the issues reported during processing and repeat the entire process again (except that you wouldn't need to map any new codes). The most common reason that this process would fail would be missing source data files. One simple option in that case is to create a dummy file with the appropriate name, but only containing the header row from your source data. This will satisfy the Automation Tool when it is looking for the file and allow the overall process to complete.

Step 5. – Update snapshots and save progress.

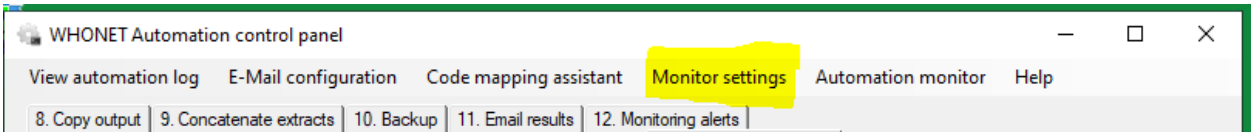
This step only requires the administrator to press a single button once again. The button will copy the newly completed production data file back to the "snapshot" folder defined in an earlier step for use in the next code mapping session.

Automation monitoring

The Automation Tool can monitor runtime and other statistics about the process when it is in fully automated mode. These data points are stored in a small database which is used by "12. Monitoring alerts" to prepare the charts and alerts associated with the alert conditions defined by the system. For example, the runtime for each individual step is recorded and the number of isolates for a given date is recorded during normal processing when the monitoring feature is enabled and configured. Using these data points we can calculate the statistics for runtimes and isolate counts for each step individually, and alert if they are higher or lower than one or two standard deviations from the mean (as specified in "12. Monitoring alerts")

Monitor settings

The monitor settings are accessible at the top of the main menu for the Automation Tool as shown below:



On this form, the user can enable the monitoring feature by checking "Enable monitor". If this is the first time you are setting up the system, you should press "Create new monitoring database" and ensure that the resulting file is selected in the "Database" text box by pressing "Browse" and locating the newly created file. Please remember to save your changes. This form typically only needs to be configured once.

Automation monitor

The interactive automation monitor interface can be accessed through the main menu for the Automation Tool as shown below:

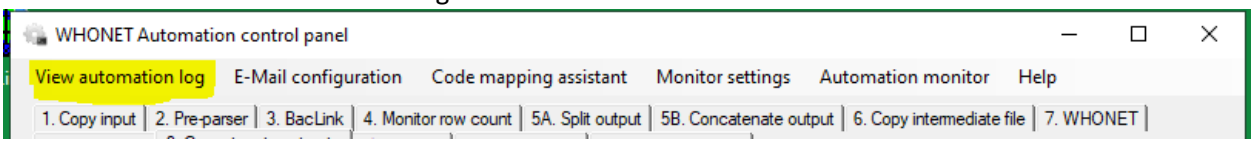
This will only work once you have data points accumulated by allowing the Automation Tool to complete its normal processing schedule for several days. After some time has passed, you can return to this form to view charts like those which are sent by email by "12. Monitoring alerts". This form allows you to interactively choose different charts to view. You can view "Runtime" or "Row count" charts.

The "Runtime" charts show the time each step took to complete in milliseconds over time. You can use the adjacent drop-down list to view each step individually, or overall on a single chart.

The "Row count" charts show you the daily isolate counts for the facilities known to the system, stratified by institution by default. You can use the adjacent drop-down list to view each facility individually, or overall on a single chart.

View automation log

This interface can be accessed through the main menu for the Automation Tool as shown below:



It displays a long list of the individual automated processing result messages. These messages can be copied somewhere else to expand the text out if needed for investigational purposes.

Manual data processing

The automation tool is still very useful even when full automation utilizing one of the various services outlined in the corresponding installation document are not used. For example, a user would still need to accomplish the same goals that the Automation Tool brings together: process the data with BacLink, combine it into a yearly file, run WHONET, copy the output to locations where their colleagues can view it. Even if this is not running by itself with the help of an automation service (for example, if you have difficulty obtaining the necessary permissions to install the service correctly), having these configuration settings here in the Automation Tool would reduce that long manual process down to a single button press. So as a worst-case scenario, you could press the "Execute date range" button once manually per day, effectively making you the automation service.

This is very close to what happens when the service is setup properly in fact. When the service "wakes up" according to its schedule or in response to a complete set of input files, it simply invokes the Automation Tool using the options you've previously saved, and with a start and end date equal to the

current date. This has the same effect as pushing the “Execute date range” button once manually per day.

Other reasons you may wish to manually process data include catching up from errors or batch processing a large amount of source data at once. For example, suppose the external process which is supposed to extract the source microbiology data from your LIS has failed to produce files for 2 weeks before being repaired, but then those files are later deposited once the system is brought back online by your IT group. The Automation Tool will not go back to reprocess these earlier files unless you have employed the “file watcher service”. Supposing that you instead used the simpler “daily schedule service”, you would need to manually process the missing data files by choosing the correct start and end dates in the Automation Tool’s interface and pressing “Execute date range”.

The purpose of these two dates and the execute button is to allow you to repeat the full sequence of configured steps once per day over the range you’ve chosen.

It is important to note that no monitoring emails will be generated during manual processing. The reason for this is that you could receive hundreds of emails potentially if a large date range were used. Instead, since you are interactively monitoring the progress, no emails are sent.

Other ways to invoke the Automation Tool

Suppose that you have difficulty obtaining the correct permissions or dedicated user account as necessary for setting up a fully automated process. As mentioned in the section above on “Manual data processing”, one option is to open the Automation Tool interactively and press “Execute date range” once per day. This will effectively replace the service with a small amount of manual work (pressing the button).

However, the Automation Tool can be invoked through the command line by passing any argument to it (the argument itself will be ignored – only its presence matters). For example, if the command “Automation Tool.exe” CMD was entered into the command prompt as shown below, the system would execute the current day’s processing just as if it had been invoked by our service. This is in fact all that the service does. It focuses solely on when to invoke the tool but does not take any other actions.

This is important because you may have another acceptable way to schedule this task without needing to user our services, which you may have a problem setting up for lack of permissions, etc.

For example, you could use the Windows task scheduler to run the command above, or another process that your IT group prefers.