Australian Government

Murray–Darling Basin Authority

Environmental Water Requirement Tool User Guide

EWR Tool Overview- Working Version

January 2026

GPO Box 1801, Canberra ACT 2601
1800 230 067
engagement@mdba.gov.au
mdba.gov.au

Published by the Murray–Darling Basin Authority
MDBA publication no: 0
ISBN (online): ISBN generated by communications
ISBN (print): ISBN generated by communications

Cataloguing data
Title: Environmental Water Requirement Tool User Guide, Murray–Darling Basin Authority Canberra, 2026. CC BY 4.0

Accessibility
The Murray–Darling Basin Authority makes its documents and information available in accessible formats. On some occasions the highly technical nature of the document means that we cannot make some sections fully accessible. If you encounter accessibility problems or the document is in a format that you cannot access, please contact us.

Acknowledgement of the Traditional Owners of the Murray–Darling Basin
The Murray–Darling Basin Authority pays respect to the Traditional Owners and their Nations of the Murray–Darling Basin. We acknowledge their deep cultural, social, environmental, spiritual and economic connection to their lands and waters.

The guidance and support received from the Murray Lower Darling Rivers Indigenous Nations, the Northern Basin Aboriginal Nations and our many Traditional Owner friends and colleagues is very much valued and appreciated.

Aboriginal people should be aware that this publication may contain images, names or quotations of deceased persons.

Version control			
Version	**Revision date**	**Author/modifier**	**Distributed to**
v.2	15/12/2025	Sirous Safari Pour	

Contents

1. Introduction

The Environmental Water Requirements (EWR) Tool is a Python-based assessment framework designed to evaluate the achievement of EWRs using daily hydrological time series. The tool supports both observed and modelled flow (and level) data and applies EWR rules as defined in states Long-Term Water Plans (LTWPs), Environmental Water Management Plans (EWMPs), and Flows Studies.

At its core, the EWR Tool uses a standalone Python package, py-ewr, to identify, assess, and summarise EWR performance at specified locations. For each site included in the input data, the tool compares hydrological time series to EWR criteria defined in inbuilt or custom tables, and calculates a range of event-based and inter-event-based statistics. These statistics are then aggregated into a set of structured outputs to support reporting, evaluation, and further analysis.

The EWR Tool is maintained as an open-source, version-controlled GitHub repository, allowing transparency in assessment logic and enabling users to run standard assessments, customise parameter inputs, or extend the tool for specific case studies. Figure 1 presents a schematic overview of the EWR Tool workflow. Information and data contained within LTWPs, EWMPs, and Flows Studies are first extracted and consolidated in collaboration with states subject matter experts, and then converted into machine-readable datasets for use by the tool. This user guide describes the structure of the EWR Tool, its input and output data, and the core modules that support EWR assessment. This User Guide is intended to support analysts, modellers, and water managers in:

- Installing and running the EWR Tool
- Preparing and supplying appropriate input data
- Understanding the internal structure and logic of the tool
- Interpreting the outputs produced by the assessment.

This guide focuses on how the tool works, what data it requires, and how to interpret its results, rather than providing policy guidance on EWR setting or LTWPs, EWMPs, and Flows Studies development.

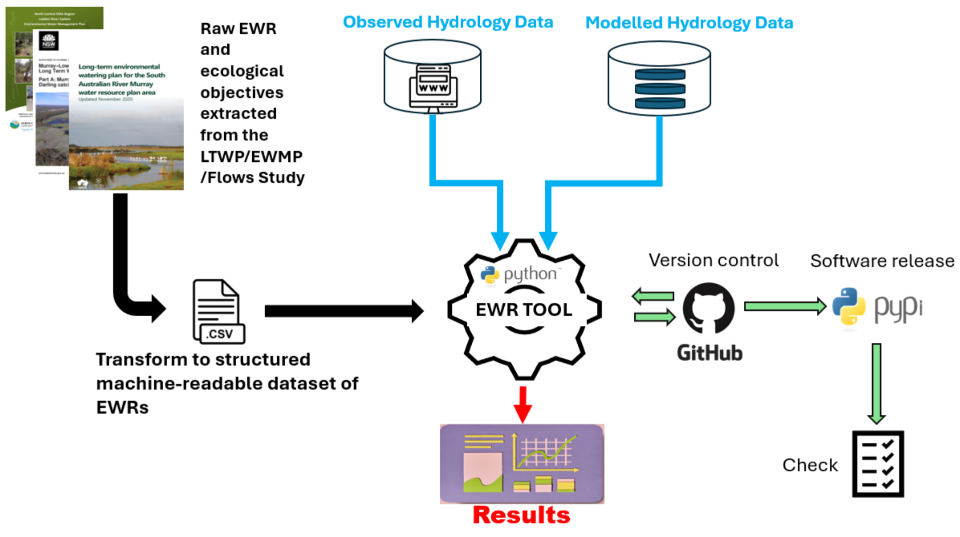

Raw EWR and ecological objectives extracted from the LTWP/EWMP /Flows Study
Observed Hydrology Data
Modelled Hydrology Data
Transform to structured machine-readable dataset of EWRs
.CSV
python
EWR TOOL
Version control
GitHub
Software release
pypi
Check
Results

Figure 1 Schematic of EWR tool workflow

2. EWR Tool installation

2.1 Python environment requirements

The EWR Tool is a Python-based application and requires access to a Python execution environment to be run. Users must ensure that a suitable Python environment is available on their local machine or chosen computing platform before installing or running the tool.

2.1.1 Minimum requirements

- Python: Version 3.9 or later is recommended
- Package management: Ability to install Python packages using pip
- File access: Read/write access to local or remote data files
- Internet access (optional): Required when running observed-mode assessments that retrieve data from online water data portals.

2.1.2 Supported python environments

The EWR Tool can be run in a range of commonly used Python environments, including:

2.1.2.1 Local python environments

- Local installation of Python on Windows, macOS, or Linux
- Virtual environments (e.g. venv, conda) are strongly recommended to isolate dependencies
- Suitable for users working with large datasets or customised parameter sheets.

2.1.2.2 Jupyter-based environments

- Jupyter Notebook/ JupyterLab
- VS Code with Jupyter extension

- Recommended for exploratory analysis, testing scenarios, and inspecting intermediate results.

2.1.2.3 Cloud-based environments

- Cloud notebook platforms that support Python and pip installations.
- These environments are suitable for users who do not wish to install Python locally, want a quick way to run analysis or demonstrate results and are working collaboratively or remotely.

2.1.3 Python environment selection guidance

- Users running standard EWR assessments with default settings can use any supported environment
- Users working with large modelled datasets, multiple scenarios, or custom tool modifications should use a local or cloud-based environment with sufficient memory and storage
- For reproducibility and stability, the use of virtual environments is recommended.

2.2 EWR Tool download

The EWR Tool is available as an open-source project on GitHub (https://github.com/MDBAuth/EWR_tool).

Users can access the repository to:

- Download the full source code.
- View README and examples run code
- Access embedded input data such as parameter sheets and metadata files.

The repository main branch contains the latest stable version of the tool. The tool requires Python 3.9 to 3.13 (inclusive). Before running it's recommended to upgrade pip by running the following code in a Python cell:

```
python -m pip install --upgrade pip
```

Users can either download the tool via PyPi or directly from GitHub main branch. To install via PyPi , run the following code in your Python cell environment:

```
pip install py-ewr
```

The version in PyPi is regularly available for latest published EWR Tool. The user also can download EWR Tool via GitHub main branch by running the following commend in Python cell:

```
pip install git+https://github.com/MDBAuth/EWR_tool.git@main
```

Users who wish to modify parameter sheets, configuration files, or core code should clone or download the GitHub repository rather than relying solely on the pip installation. If you encounter any issues with the package, you can either submit an issue through the GitHub Issues section of the repository or contact the development team using the provided email address on the README.

2.2 Running the EWR Tool in observed mode

Observed mode is used to assess EWR achievement using observed hydrological data sourced directly from states water data portals. In this mode, the EWR Tool automatically retrieves flow or level data for selected gauges and dates, then applies the EWR assessment logic using the embedded parameter and metadata tables. Observed data retrieval is handled by a companion Python package, MDBA Gauge Getter, which provides a unified interface to access surface water data across multiple state water portals and the Bureau of Meteorology (BoM) water portal. For more information on data sources and functionality, users can refer to the MDBA Gauge Getter GitHub repository.

2.1.4 Required user inputs

To run the tool in observed mode, users must specify:

- A date range covering the assessment period
- A list of gauge numbers to be assessed.

Dates should be supplied using Python datetime objects and should align with whole water-years where possible (1 July to 30 June).

2.1.4.1 Example: running observed mode

```
from datetime import datetime


# USER INPUT REQUIRED >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


dates = {

    'start_date': datetime(YYYY, 7, 1),

    'end_date': datetime(YYYY, 6, 30)

}


gauges = ['Gauge1', 'Gauge2']


# END USER INPUT <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


from py_ewr.observed_handling import ObservedHandler


# Initialise the observed mode handler

ewr_oh = ObservedHandler(gauges=gauges, dates=dates)
```

Once initialised, the ObservedHandler will download observed data for the specified gauges and dates using MDBA Guage-Getter and applies data quality filtering. The data then are linked to EWR codes matching the specified gauges defined in the parameter sheet embedded in the tool. The tool then performs the EWR assessment using the supplied flow or level time series.

2.1.4.2 Generating EWR outputs

After the tool has run, users can generate the full set of EWR outputs using the following command in Python cell:

```
# Table 1: Summary EWR results for the entire time series

summary_results = ewr_oh.get_ewr_results()


# Table 2: EWR results aggregated to water years

yearly_results = ewr_oh.get_yearly_results()


# Table 3: All detected events (no duration requirement)

all_events = ewr_oh.get_all_events()


# Table 4: Inter-event periods between all detected events

all_interEvents = ewr_oh.get_all_interEvents()


# Table 5: Successful events meeting duration requirements

all_successfulEvents = ewr_oh.get_all_successful_events()


# Table 6: Inter-event periods between successful events

all_successful_interEvents = ewr_oh.get_all_successful_interEvents()


# Optional: export results to CSV files

summary_results.to_csv('summary_results.csv')

yearly_results.to_csv('yearly_results.csv')

all_events.to_csv('all_events.csv')

all_interEvents.to_csv('all_interEvents.csv')

all_successful_Events.to_csv('all_successful_events.csv')

all_successful_interEvents.to_csv('all_successful_interEvents.csv')
```

Each method returns a Pandas DataFrame, which can be exported directly to CSV for reporting or analysed further.

2.1.4.3 Notes and considerations

- Observed mode relies on the availability and quality of data from state water portals; missing or poor-quality data may affect results
- Gauge numbers must be present in the tool's embedded site metadata tables to be linked to EWR definitions
- For assessments requiring customised parameters, users should modify the parameter sheets and configuration files within a local copy of the GitHub repository.

2.4 Running the EWR Tool in modelled scenario mode

Modelled scenario mode is used to assess EWR achievement using modelled flow or level time series, such as hydrological or climate scenario outputs. In this mode, users supply one or more model files, specify the model format, and run the EWR assessment directly on those datasets.

Each scenario may consist of one or multiple model files. The EWR Tool processes each file individually and then combines the results at the scenario level.

2.4.1 Required user inputs

To run the tool in modelled scenario mode, users must specify:

- Scenario definitions: A dictionary where the key is the scenario name, and the value is a list of file paths for the model output files associated with that scenario
- Model format: The format of the supplied model files. This determines how the tool reads headers, dates, and gauge identifiers.

File paths can be relative (to the working directory) or absolute paths pointing to local or remote locations.

2.4.1.1 Supported model formats

The EWR Tool currently supports the following model formats:

- Bigmod - MDBA
- Source - NSW (res.csv)
- FIRM - MDBA
- Standard time-series
- ten thousand year

If a model format is not supported, users should contact the MDBA to discuss extending compatibility.

2.4.1.2 Example: running modelled scenarios

```
# USER INPUT REQUIRED >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


# Minimum of one scenario and one associated file required

scenarios = {

    'Scenario1': ['path/to/file1.1', 'path/to/file1.2', 'path/to/file1.3'],

    'Scenario2': ['path/to/file2.1', 'path/to/file2.2', 'path/to/file2.3']

}

model_format = 'Bigmod - MDBA'


# END USER INPUT <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Scenario names are incorporated into the “Scenario” column of the output tables and are also used when exporting results to CSV. It is therefore recommended that scenario names are informative and traceable to the underlying model configuration.

2.4.1.3 Generating EWR outputs

To generate output results run the following code in a Python cell and save the results in a desired format and path.

```
from py_ewr.scenario_handling import ScenarioHandler

import pandas as pd

summary_results_dict = {}

yearly_results_dict = {}

all_events_dict = {}

all_interEvents_dict = {}

all_successful_Events_dict = {}

all_successful_interEvents_dict = {}

for scenario_name, scenario_list in scenarios.items():

    summary_results = pd.DataFrame()

    yearly_results = pd.DataFrame()

    all_events = pd.DataFrame()

    all_interEvents = pd.DataFrame()

    all_successful_Events = pd.DataFrame()

    all_successful_interEvents = pd.DataFrame()


    for file in scenario_list:

        # Run the EWR Tool for each model file
        ewr_sh = ScenarioHandler(
            scenario_file=file,
            model_format=model_format
        )

        # Combine results from multiple files within the same scenario
        summary_results = pd.concat(
            [summary_results, ewr_sh.get_ewr_results()], axis=0
        )
        yearly_results = pd.concat(
            [yearly_results, ewr_sh.get_yearly_ewr_results()], axis=0
        )
        all_events = pd.concat(
            [all_events, ewr_sh.get_all_events()], axis=0
        )
        all_interEvents = pd.concat(
            [all_interEvents, ewr_sh.get_all_interEvents()], axis=0
        )
        all_successful_Events = pd.concat(
            [all_successful_Events, ewr_sh.get_all_successful_events()], axis=0
        )

        all_successful_interEvents = pd.concat(
            [all_successful_interEvents,
            ewr_sh.get_all_successful_interEvents()], axis=0
        )

    # Optional: export results to CSV files
    summary_results.to_csv(scenario_name + '_summary_results.csv')
    yearly_results.to_csv(scenario_name + '_yearly_results.csv')
    all_events.to_csv(scenario_name + '_all_events.csv')
    all_interEvents.to_csv(scenario_name + '_all_interEvents.csv')
    all_successful_Events.to_csv(scenario_name + '_all_successful_events.csv')
    all_successful_interEvents.to_csv(
        scenario_name+'_all_successful_interEvents.csv'
    )

    # Optional: store results in dictionaries for further processing
    summary_results_dict[scenario_name] = summary_results
    yearly_results_dict[scenario_name] = yearly_results
    all_events_dict[scenario_name] = all_events
    all_interEvents_dict[scenario_name] = all_interEvents
    all_successful_Events_dict[scenario_name] = all_successful_Events
    all_successful_interEvents_dict[scenario_name] = all_successful_interEvents
```

2.1.4.4 Notes and considerations

- All model files must comply with the selected model format requirements
- When loading hydrology model data from "Bigmod - MDBA", "Source - NSW (res.csv)", or "FIRM - MDBA", gauge or node identifiers in model outputs must be present in the tool's embedded SITEID metadata tables to be linked to EWR definitions
- When multiple files are supplied for a scenario, results are concatenated to form a single scenario-level output.

3. Input data for EWR Tool

The EWR Tool requires two broad categories of input data. One category is embedded within the tool but can be modified for specific case studies, while the other must be provided by the user for each analysis.

3.1Embedded input data

The embedded input data include the parameter sheet, the objective reference, Site IDs and Config file. These datasets are distributed with the tool and support the standard EWR assessment framework.

3.1.1 Parameter sheet

Each time the EWR Tool is run, it reads the parameter sheet, which is stored as a CSV file within the tool. The parameter sheet contains EWR parameters defined in states LTWPs. The parameter sheet can be downloaded from the GitHub repository. Users who wish to explore alternative rule sets or test different parameter configurations are encouraged to download the parameter sheet to their local machine or individual repositories and apply changes there. Modified parameter sheets can run through the tool for bespoke analyses or be proposed for inclusion in the main GitHub branch, where requests will be assessed by the MDBA team on a case-by-case basis. Bespoke parameter sheets can be passed to the EWR tool by adding the "parameter_sheet" argument to the main function as below:

```
# USER INPUT REQUIRED >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

ewr_sh = ScenarioHandler(

    scenario_file=file,

    model_format=model_format,

    parameter_sheet='path/to/modified/parameter_sheet.csv'

)
```

3.1.2 Objective reference

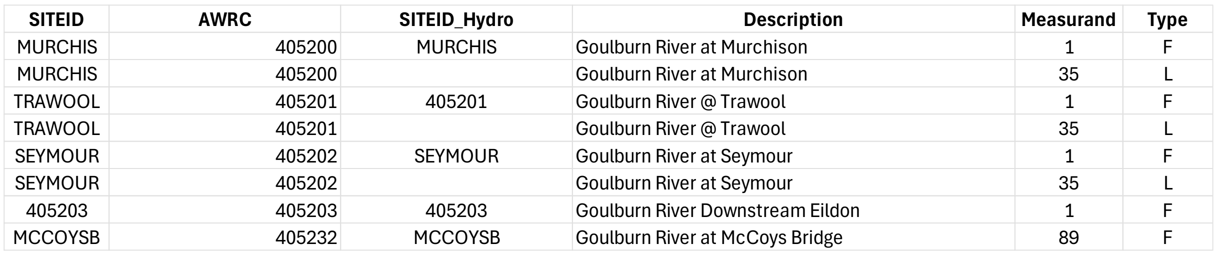
The objective reference sheet is stored as a CSV file within the tool, alongside the parameter sheet. This dataset contains ecological objectives and targets as defined in LTWP or management frameworks. The objective reference is used to map ecological objectives to each EWR parameter and is primarily applied during post-processing of EWR results.

3.1.3 Site ID

The Site ID file is used for the model file types:

- "Bigmod - MDBA"
- "Source - NSW (res.csv)"
- "FIRM - MDBA"

The Site ID files are stored as a CSV file within the tool under the model_metadata directory inside the py-ewr package. The input data has the time-series of flows (or levels) associated to a given model node. During processing, the EWR Tool searches each model node for a match in the "Site ID" column. If a match is found, then the item in the "AWRC" column will determine the gauge number, and the "Type" column will inform if it is flow or level (as illustrated in Figure 2). If the node of the input data is not in the Site ID reference file, then no EWR assessment will be performed for that node.



SITEID	AWRC	SITEID_Hydro	Description	Measurand	Type
MURCHIS	405200	MURCHIS	Goulburn River at Murchison	1	F
MURCHIS	405200		Goulburn River at Murchison	35	L
TRAWOOL	405201	405201	Goulburn River @ Trawool	1	F
TRAWOOL	405201		Goulburn River @ Trawool	35	L
SEYMOUR	405202	SEYMOUR	Goulburn River at Seymour	1	F
SEYMOUR	405202		Goulburn River at Seymour	35	L
405203	405203	405203	Goulburn River Downstream Eildon	1	F
MCCOYSB	405232	MCCOYSB	Goulburn River at McCoys Bridge	89	F

Figure 2 Example of SITE ID sheet in the tool

3.1.4 Configuration file (ewr_calc_config)

The configuration file is stored as a JSON file within the parameter_metadata directory inside the py_ewr package. This file is directly linked to the parameter sheet and other internal modules of the EWR Tool and controls how individual EWRs are interpreted during processing. Each EWR code defined in the parameter sheet must also be declared in the configuration file for it to be included in the calculations. The configuration file specifies both the EWR code and the type of gauge it applies to (for example, flow or level). For instance, an EWR with the code IC1_S, which is assessed against flow data, is defined in the configuration file as IC1_S-single-F. Users who add new EWRs to the parameter sheet must also add corresponding entries to the configuration file. If an EWR is not defined in the configuration file, it will not be assessed by the tool.

3.2 User-provided input data

3.2.1 Observed data

When using observed flow data, the EWR Tool retrieves hydrological data directly from relevant state and national water data portals for the selected gauges and time period. This functionality is provided through the MDBA Gauge Getter Python package, which is included as part of the EWR Tool. The observed data are retrieved from the following publicly available portals:

- New South Wales: https://realtimedata.waternsw.com.au
- Queensland: https://water-monitoring.information.qld.gov.au
- Victoria: https://data.water.vic.gov.au
- South Australia (national source): https://www.bom.gov.au/waterdata
- Australian Capital Territory: https://www.bom.gov.au/waterdata.

By default, the tool retrieves daily mean flow data (ML/day) for each gauge. Where available and applicable, other variables such as water level or storage volume, as well as different temporal resolutions and aggregation methods, may also be retrieved.

Handling Data Access Issues: In some situations, users may encounter errors or access issues when retrieving observed data from state or national portals (for example, due to temporary outages, network restrictions, or changes to portal services). If this occurs, users have the following options:

- Retrieve data from alternative source
- Users with appropriate Python expertise may modify the tool to retrieve observed data directly from alternative services, such as BoM Water Data Online, and supply the data to the tool
- Download observed data and run it through the tool using the scenario assessment functionality

- Users may manually download observed data (for example, from BoM or state portals), format it as a supported scenario input type (such as the Standard time-series format), and run the assessment using the scenario mode of the EWR Tool. This allows observed data to be assessed without relying on live data retrieval during execution.

These approaches provide flexibility and ensure that EWR assessments can continue even when direct portal access is unavailable.

3.2.2 Modelled scenario data

When using modelled scenario data, users must specify the file locations of the relevant model output files. The EWR Tool requires these files to conform to supported data structures and datetime formats. If a model format (including datetime variations) falls outside the supported specifications, users should contact the developers listed in the EWR Tool GitHub repository to discuss potential extensions to the tool. Failure to meet the required formatting standards may result in data ingestion or processing errors. The py-ewr package is currently compatible with the following model formats:

- Bigmod – MDBA
- Source – NSW (res.csv)
- FIRM – MDBA
- ten thousand year
- Standard time-series

If you are working with modelled scenario data formats not listed above, please contact the MDBA using the email address on the README file. The development team can assess the format and, where appropriate, update the codebase to support additional model structures.

Appendix A provides description of input model data type.

4. EWR tool structure

The EWR Tool repository (publicly accessible at https://github.com/MDBAuth/EWR_tool) contains the source code, configuration files, example scripts, and test suites required to implement, run, and validate the tool's functionality. The overall structure of the repository is shown below.

```
EWR_tool/                     # Repository root

.github/workflows             # GitHub workflows

py_ewr/                       # Core Python package

tests/                        # Python test scripts

unit_testing_files/           # Supporting test data

example_parallel_run.py       # Example usage script for parallel computation

EWR_tool_requirements.txt     # Package dependencies

requirements-dev.txt          # Dev/test dependencies

setup.py                      # Package build/install script

pyproject.toml                # Build metadata

README.md                     # Project overview and instructions

LICENSE                       # Open-source license
```

The core packages are in the py_ewr folder. Input data handling is carried out by data_inputs.py, processing dates and gauges information, flow handling are found in evaluate_EWRs.py. The observed_handling.py used to pull all relevant gauge data for EWR processing for observed mode of calculation. For model scenario calculations, scenario_handling.py is used to read appropriate model format data and calculations. summarise_results.py converts the results into the six different output formats.

4.1 Module data_inputs.py

This module is responsible for loading, validating, standardising, and managing all EWR parameter, configuration, and metadata inputs used by the EWR Tool.

4.1.1 Configuration management

- Loads the EWR configuration file (ewr_calc_config.json) from the tool's internal parameter_metadata directory or from a user-specified path
- Defines how EWR codes are interpreted (e.g. flow vs level EWRs)

- Ensures only EWRs defined in both the parameter sheet and configuration file are assessed.

4.1.2 Parameter sheet ingestion and cleaning

- Loads the EWR parameter sheet (parameter_sheet.csv) and extracts only the columns required for assessment
- Separates valid EWRs from those that cannot be assessed (e.g. missing thresholds, missing duration)
- Applies default values where required to ensure calculations can proceed
- Converts parameters into consistent numeric data types (integers and floats)
- Splits combined month/day values into separate fields where required.

The output of this process is:

- a cleaned EWR parameter table used for assessment
- a table of excluded EWRs for quality assurance and reporting.

4.1.3 Parameter mapping and component selection

- Defines a mapping dictionary that translates parameter sheet column names into internal variable names
- Identifies which parameter components are required for each EWR assessment category (e.g. flow, level, cumulative, barrage, multi-gauge).

4.1.4 Model metadata and gauge mapping

Loads Site ID mapping files for different modelling frameworks, including: MDBA BigMod, MDBA FIRM, NSW Source. Links model node names to AWRC gauges (gauge number or node name), allowing modelled data to be assessed against the correct EWRs. Supports multi-gauge EWRs, where flows from multiple gauges must be combined. If a node or gauge is not found in the relevant metadata file, the EWR Tool cannot associate the data with an EWR and no assessment is performed.

4.1.5 Gauge classification and grouping

- Identifies and groups gauges by type, including flow gauges, level gauges, lake level gauges
- Includes special handling for: weirpool gauges, barrage gauges, states-specific gauges (NSW, QLD, VIC).

4.1.6 Quality control and utilities

- Defines poor-quality data codes that should be excluded from analysis
- Identifies whether a weirpool EWR represents a raising or falling event
- Provides helper functions to retrieve planning unit identifiers and names, individual EWR parameter values and gauges processed across scenarios.

4.1.7 Ecological objective mapping

- Merges the EWR parameter sheet with the objective reference file to produce a long-form table linking: EWRs, planning units, gauges, ecological objectives and targets
- Supports post-processing, reporting, and alignment with LTWPs.

This module ensures that: EWR definitions are internally consistent, model outputs can be reliably linked to gauges, only valid and fully defined EWRs are assessed, and downstream calculation modules receive clean, standardised inputs.

4.2 Module evaluate_EWRs.py

The evaluate_EWRs.py module is the core assessment engine of the EWR Tool. It applies EWR rules to processed flow and level time-series data and determines whether each EWR has been achieved for a given gauge, planning unit, and time period.

Primary purpose of this module:

- evaluating daily flow and level data against EWR rules defined in the parameter sheet and configuration file
- producing event-based and summary statistics used in EWR reporting.

4.2.1 Temporal filtering and water-year handling

The module:

- filters daily data to the relevant seasonal windows (month and day ranges),
- groups results by water-year
- tracks event continuity, gaps, and inter-event tolerances in accordance with EWR definitions.

4.2.2 EWR rule application

evaluate_EWRs.py implements the logic required to assess a wide range of EWR types, including:

- flow-based EWRs (e.g. freshes, low flows, cease-to-flow)
- cumulative volume EWRs
- level and lake-level EWRs
- weirpool raising and falling events
- barrage flow and barrage level EWRs
- multi-gauge EWRs
- specialised EWRs such as nesting, water stability, and rise/fall constraints.

Each EWR type is evaluated using the appropriate thresholds, durations, frequencies, and timing windows defined in the parameter sheet. Standard rules shared by the majority of EWRs are described in Appendix B.

4.2.3 Event detection and output generation

For each applicable EWR, the module:

- detects qualifying flow or level events
- measures event duration, frequency, and spacing.

The module produces structured outputs that feed directly into the summarise_results.py module, which will then produce the standard outputs. This includes the event-level results as well as the achievement of EWR's by water year for each planning unit. These outputs are returned as Pandas DataFrame's and are used to form the six standard result tables produced by the EWR Tool.

4.2.4 Overview

Within the overall workflow, evaluate_EWRs.py:

- receives cleaned input data and validated parameters from upstream modules
- applies EWR rules and performs event-based analysis
- returns structured results for post-processing by the summarise_results.py module.

4.3 Module summarise_results.py

The summarise_results.py module is responsible for post-processing, aggregation, and summarising EWR assessment results produced by the EWR evaluation engine. It converts detailed, year-by year EWR outputs and event records into summary tables suitable for reporting, comparison, and interpretation. This module is used for both observed and modelled scenario analyses. Primary purpose of this module:

- aggregates yearly EWR assessment results across the full analysis period
- derives summary performance metrics for each EWR, gauge, and planning unit
- summarises event and inter-event behaviour
- produces final, tidy result tables aligned with EWR reporting requirements.

4.3.1 Transformation of yearly results

The module restructures detailed yearly EWR outputs from a nested dictionary into a Pandas DataFrame, where each row represents:

- a single EWR
- at a specific gauge
- within a planning unit
- for a given scenario.

4.3.2 Aggregation of EWR performance metrics

For each EWR, the module calculates summary statistics across all years, including:

- number of years with qualifying events

- frequency of EWR achievement
- total and average number of events
- total and average event days
- achievement counts and achievement per year
- number of missing data days and total assessed days.

4.3.3 Event statistics and event-level outputs

Using detailed event records generated during evaluation, the module:

- counts the total number of events
- calculates total event days and average event length
- optionally produces event-level tables containing start and end dates, event duration, and water-year attribution.

4.3.4 Inter-event period analysis

The module derives inter-event periods (time between qualifying events) by:

- identifying gaps between events
- calculating inter-event durations
- determining the maximum rolling inter-event period for each EWR and year.

4.3.5 Success filtering

For event-based analyses, the module can:

- filter events based on minimum duration (minimum spell)
- retain only successful events
- exclude partial or non-qualifying events from summaries.

4.3.6 Integration with EWR parameters

The summary outputs are joined with selected fields from the EWR parameter sheet, including:

- target frequency (percentage)
- maximum recommended inter-event period (years)
- planning unit and SDL names
- state and multigauge indicators.

4.3.7 Final summary output

The primary output of this module is a final EWR summary table, which:

- contains one row per scenario, gauge, planning unit
- and EWR includes achievement, frequency, event, and inter-event metrics, and is returned as a Pandas DataFrame for reporting, export, or visualisation.

4.3.8 Overview

Role in the EWR Tool Workflow within the overall EWR Tool workflow, summarise_results.py:

- receives detailed yearly results and event records from the evaluation modules
- aggregates results across the full assessment period
- derives event and inter-event statistics
- produces final summary tables aligned with EWR reporting needs.

This module is the final analytical step before results are presented to users or exported for further analysis.

4.4 Module observed_handling.py

The observed_handling.py module manages the ingestion and assessment of observed (gauged) flow and water level data within the EWR Tool. It provides the complete workflow required to retrieve observed data from states water portals and convert it into EWR assessment results. This module is only used when the observed data option is selected. Primary purpose of this module:

- retrieves observed flow and level data from state water data portals
- cleans and standardises the time series
- assigns gauges to the correct EWR types
- evaluates EWR achievement using the same assessment logic applied to modelled data.

4.4.1 Retrieval of observed data

Observed data are retrieved using the MDBA Gauge Getter Python package, which provides a unified interface to multiple state water data portals. For more information about Gauge-Getter please visit (https://github.com/MDBAuth/MDBA_Gauge_Getter). The module automatically:

- requests daily data for the user-defined date range
- retrieves flow, level, and lake level data as required
- abstracts away states-specific data source differences.

4.4.2 Gauge classification

The module classifies user-specified gauges into:

- flow gauges,
- river level gauges
- lake level gauges
- based on information stored in the EWR parameter sheet and internal metadata tables.

Special handling is applied for the following EWR types to allow for the searching of additional hard coded gauges to support their evaluation:

- weirpool gauges (which may require both flow and level data)
- Coorong, Lower Lakes, and Murray Mouth (CLLMM) EWRs

- multi-gauge EWRs.

4.4.3 Data cleaning and standardisation

Retrieved data are cleaned and converted into a consistent format by:

- removing poor-quality observations using predefined quality-control codes
- converting values to numeric types
- aligning all gauges to a common daily date index
- restructuring the data so that each gauge appears as a single column.

4.4.4 Run The workflow to generate results

Now we have the correctly structured input data, the observed_handling.py module will use the relevant functionality from data_inputs.py, then evaluate_EWRs.py, then summarise_results.py, to complete the workflow and generate all the outputs for further analysis, reporting, or visualisation.

Full descriptions are provided in "4.1 Module data_inputs.py", "4.2 Module evaluate_EWRs.py" and "4.3 Module summarise_results.py", the observed_handling.py module utilises these modules to:

- run the assessment of the flow and level Pandas DataFrame's and return the EWR achievement as well as event information
- convert this information into the standard six output formats to generate event-level, yearly, and summary EWR results specified in "5 Outputs from the EWR tool".

4.4.5 Overview

Role in the EWR Tool Workflow, within the overall workflow, observed_handling.py:

- retrieves and cleans observed gauge data
- classifies gauges and assigns EWR types
- performs EWR evaluation using the core assessment engine
- generates event-level, yearly, and summary EWR results.

This module ensures that observed flow and level data are assessed in a manner that is consistent with modelled scenario assessments.

4.5 Module scenario_handling.py

The scenario_handling.py module manages the ingestion, standardisation, and assessment of modelled scenario flow and level data within the EWR Tool. It provides the workflow required to read model outputs from supported hydrological models, map model nodes to gauges, and evaluate EWRs consistently across scenarios. This module is used when the modelled scenario option is selected. The primary purpose of this module is:

- reads modelled time-series data from supported model formats
- maps model nodes to gauges using internal metadata file (relevant Site ID files)
- standardises data structures across models and scenarios

- performs EWR assessments using the core evaluation engine.

4.5.1 Scenario file ingestion

The module ingests modelled data files provided by the user and supports multiple model formats, including:

- MDBA BigMod
- MDBA FIRM
- NSW Source (res.csv)
- 10,000-year climate sequences
- Standard time-series formats.

4.5.2 Multi-scenario processing

The module supports:

- assessment of multiple scenarios in a single run
- separation of results by scenario name
- aggregation of EWR results across scenarios for comparison and reporting.

4.5.3 Model metadata and gauge mapping

Model node names are matched to AWRC gauges using metadata files stored within the EWR Tool (SITEID). This step ensures that modelled flows (or levels) are assessed against the correct EWRs defined in the parameter sheet. If a model node cannot be matched to a gauge, then no EWR assessment is performed for that node, and the issue is reported to the user.

4.5.4 Data standardisation

Modelled data are converted into a consistent internal format by:

- standardising datetime formats
- aligning data to a daily time step
- ensuring numeric data types
- restructuring the data so each gauge is represented as a single column.

4.5.5 Run the workflow to generate results

With the correctly structured input data the same steps as observed_handling.py will be run. The scenario_handling.py module will use the relevant functionality from data_inputs.py, then evaluate_EWRs.py, then summarise_results.py, to complete the workflow and generate all the outputs for further analysis, reporting, or visualisation.

Full descriptions are provided in “4.1 Module data_inputs.py”, “4.2 Module evaluate_EWRs.py” and “4.3 Module summarise_results.py”, the observed_handling.py module utilises these modules to:

- run the assessment of the flow and level Pandas DataFrame’s and return the EWR achievement as well as event information

- convert this information into the standard six output formats to generate event-level, yearly, and summary EWR results specified in “5 Outputs from the EWR tool”.

4.5.6 Logs generation

There is extra functionality to generate logs which contain information useful for quality assurance and testing of the run through. For each EWR listed in the parameter sheet, the following checks are performed during the run and can be written to a csv file for further analysis:

- was there a result returned in summary results? (which checks if each EWR had any issues and got through all stages)
- do the gauges for each EWR match a node in the siteID file?
- was that gauge found in the model file?
- was that measurand also found in the model file? (meaning if an EWR requires flow at gauge 425125, then you must have both the correct gauge and flow contained in the hydrology model file)
- was there a calc_config match found? (this pairs each EWRCode i.e. BF1 with a calculation type that will inform what EWR calculation to perform)
- any additional siteID info. Note that in the SiteID file it is possible to have multiple different SiteID nodes that all match to the same gauge. These are required as different input data formats will have different names of nodes to represent the same gauge. The “additional info” that the logs check is which of these SiteID nodes did it match with for this run through, as well as list the extra SiteID’s that it did not match with but represent the same gauge.

4.5.7 Overview

Role in the EWR Tool Workflow within the overall workflow, scenario_handling.py:

- ingests and standardises modelled scenario data
- links model outputs to EWR definitions via gauge mapping
- evaluates EWR achievement using the core assessment engine
- produces structured results for comparison across scenarios.

This module enables consistent, transparent, and repeatable EWR assessments across multiple modelling scenarios.

4.6 Unit testing in the EWR Tool

Unit testing is necessary component of the EWR tool’s quality assurance framework. The purpose of unit testing is to verify that individual components of the tool behave as expected, produce consistent outputs, and continue to function correctly as the codes inside the tool expands and evolves. By systematically testing core modules inside the tool, the unit testing help ensure the reliability, transparency and reproducibility of the EWR results outputs.

All unit tests of the EWR Tool are located in the tests directory of the GitHub repository. These tests are designed to mirror the workflows used in running observed and scenario mode assessments.

4.6.1 Structure of the unit tests

The unit testing modules inside the test folder are as follows:

- conftest.py: Contains shared fixture for configuration across multiple test files
- test_data_input.py: Tests validating the input data, including time series formatting, required fields, checking consistency for input data module ensuring correct interpretation of input data before assessments
- test_evaluate_ewrs.py: Validates the evaluate module logic against known input conditions, including thresholds, durations rules
- test_evaluate_ewr_rest.py: This is supplementary file for test_evaluate_ewr.py
- test_observed_handling.py: Tests the observed module workflow, including interaction with retrieved gauge data and EWR table outputs
- test_scenario_handling.py: Tests the scenario module workflow including ingestion of model files and multiple scenario handling
- test_summarise_results.py: Tests the summary module workflow and aggregation logic to produce file EWR results tables.

4.6.1.1Test data files

The unit_testing_files directory contains all static files required to run the unit tests. These include examples of time series data, model outputs and expected results outputs. Using fixed and controlled input files allows the tests to produce expected outputs, making it possible to detect unintended changes in behaviour when the code is modified.

Role of Unit Testing in summary:

- To ensure EWR rules correctly implemented as defined in the parameter sheets and code logic
- To ensure changes to the code do not alter assessment outcomes unintentionally
- To ensure observed and scenario workflow produce consistent outputs
- To ensure output tables and statistics remain stable.

4.6.1.2 Outputs from the EWR Tool

After an EWR Tool run is completed, the tool generates six output tables, which can be exported as CSV files or accessed directly as Pandas DataFrame’s for further analysis. These outputs provide comprehensive information on EWR achievement, event behaviour, and inter-event periods, based on the rules defined in the EWR parameter sheet. All outputs are downloadable as CSV files or available as Pandas DataFrame’s as shown in Figure 3. The six standard outputs are listed below.

summary_results: Provides EWR performance statistics over the full assessment period, including achievement, event counts, and frequency metrics for each EWR, gauge, planning unit, and scenario.

yearly_results: Contains year-by-year EWR assessment results, showing whether each EWR was achieved in each water-year and summarising annual event statistics.

all_events: Records all detected events where flow, level, or volume thresholds are met within the specified timing window, regardless of whether duration or frequency requirements are satisfied.

all_successful_events: Includes only events that fully meet all EWR requirements, including duration and any additional constraints defined in the parameter sheet.

all_interEvents: Reports all inter-event periods (gaps between consecutive events), including those associated with both successful and non-successful events.

all_successful_interEvents: Reports inter-event periods calculated only between successful events, which are used to assess compliance with maximum inter-event requirements.

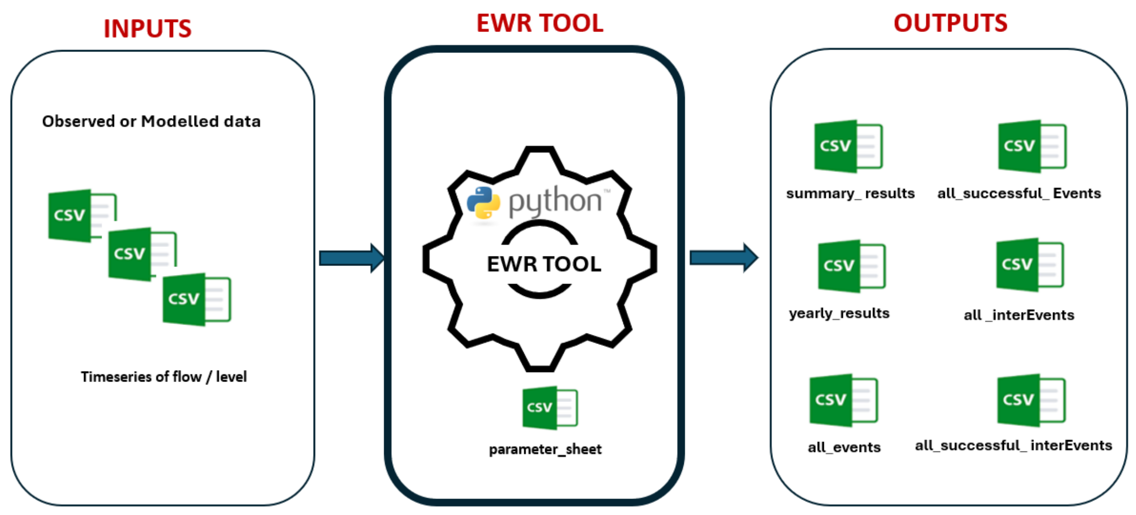


Figure 3 Schematic of EWR Tool output results

The following sections describe the six standard output datasets generated by the EWR Tool. For each output, the purpose of the dataset is explained, followed by a detailed description of the variables it contains.

4.7 Summary results output (summary_results)

The summary_results table provides aggregated EWR performance metrics over the full assessment period for each scenario, gauge, planning unit, and EWR. Each row represents one unique “Scenario”, “Gauge”, “PlanningUnit” and “Code” combination.

4.7.1 General columns

Scenario: Name or type of the scenario analysed. This identifies whether results are based on observed data or a specific modelled scenario.

Gauge: AWRC gauge number (in SiteID file) used to assess the EWR. This is the primary gauge against which flow or level data are evaluated.

PlanningUnit: Name of the planning unit as defined in the LTWP, EWMP, or Flows study. This links the EWR result to its management context.

State: Name of state or territory which the EWR code belongs to.

SWSDLName: Surface Water SDL Resource Units name where the gauge and code is related to.

Code: EWR code as defined in the parameter sheet (for example, BF1_a). Each code represents a distinct ecological flow or level requirement.

Multigauge: Gauge number of the second gauge used in assessment where the EWR requires flows from more than one gauge to be combined. This column is populated only for multi-gauge EWRs; otherwise, it is empty.

NoDataDays: Total number of days for which hydrological data were missing or unavailable in the input time series. These days are excluded from event detection and EWR assessment calculations.

TotalDays: Total number of days assessed for the EWR across the entire input time series. This value is derived from the yearly results and represents the maximum possible number of days available for EWR evaluation, excluding data gaps.

4.7.2 EWR achievement metrics columns

EventYears: Number of years in which the EWR was fully achieved. A year is counted only when all annual EWR requirements are met, including flow/level/volume thresholds, timing windows, duration, and required number of events per year.

Frequency: Percentage of years in which the EWR was achieved, calculated as: *EventYears* divided by total number of years in the data multiply by 100 to get percentage. The Frequency for cease-to-flow (CTF) EWRs uses *maxRollingAchievement* in the yearly results; for all other EWRs, it is based on *EventYears.*

TargetFrequency: Target frequency specified in the LTWPs, expressed as a percentage of years. This represents the management objective against which the calculated Frequency is compared**.**

AchievementCount: Total number of successful EWR achievements across the entire time series. This accounts for EWRs that require multiple successful events within a single year, with each successful achievement counted separately**.**

AchievementPerYear: Average number of achievements per year, calculated as: *AchievementCount* divided by total number of years in the input time series.

4.7.3 Event statistics columns

EventCount: Total number of successful EWR events across the entire time series. This is the sum of *numEvents* from the yearly results and includes only events that meet all EWR requirements (thresholds, timing, and duration).

EventCountAll: Total number of all detected events, regardless of success. This includes any event that meets the flow/level/volume threshold and timing window, even if duration or other requirements are not met**.**

EventPerYear: Average number of successful events per year, calculated from *numEvents* in the yearly results.

EventPerYearAll: Average number of all detected events per year, calculated from *numEventsAll* in the yearly results. This includes both successful and unsuccessful events.

AverageEventLength: Average duration, in days, of all detected events that meet the EWR flow, level, or volume threshold and timing window. This includes both successful and unsuccessful events and provides an indication of typical event persistence once threshold conditions are met.

ThresholdDays: Total number of days across the entire assessment period where the flow, volume, or water level exceeds the EWR threshold within the specified timing window. This value is derived from the *all_events* table and includes days associated with events that do not necessarily meet minimum duration or frequency requirements.

MaxInterEventYears: Target maximum inter-event period, expressed in years, as specified in the LTWP. This represents the longest allowable gap between successful EWR events and is used to assess inter-event compliance.

4.7.4 Interpretation notes

- Use Frequency and TargetFrequency to assess whether an EWR meets its long-term planning objective
- Use AchievementCount and AchievementPerYear to understand how often EWRs requiring multiple events per year are satisfied
- Use EventCountAll and EventsPerYearAll to investigate partial successes
- Use Multigauge to identify EWRs that depend on combined flows from multiple gauges
- ThresholdDays is useful for understanding how often threshold conditions occur, even when full EWR achievement is not reached
- NoDataDays allows for quality assurance of individual EWR results to identify if we have limited data available for the corresponding gauge.

4.8 Yearly results output (yearly_results)

The yearly_results table reports annual EWR performance for each scenario, gauge, planning unit, and EWR. Results are calculated on a water-year basis (1 July to 30 June) and provide detailed insight into event behaviour, achievement status, and data availability for each year. Each row represents one "Year", "Scenario", "Gauge", "PlanningUnit" and "Code" combination.

4.8.1 General columns

Scenario, Gauge, PlanningUnit, State, SWSDLName, Code, Multigauge (same as above).

Year: Water-year to which the event is assigned. Water-years run from 1 July to 30 June. Events in all_events are assigned to the year in which they finish. For example, an event starting on 24 October 2022 and ending on 30 April 2023 is assigned to water-year 2023. This convention ensures that full event characteristics (including total duration) are associated with a single year.

missingDays: Number of days in the year where hydrological data were unavailable or removed due to poor quality. Poor quality data are identified using quality codes.

totalPossibleDays: Total number of days available in the hydrological time series for the year (typically 365 or 366 days), including days later excluded due to data quality filtering.

4.8.2 Event statistics columns

eventYear: Indicates whether the EWR was achieved in the year. A value of 1 means all required EWR components were met (thresholds, timing, duration, and required number of events); 0 indicates the EWR was not achieved. For CTF EWRs (except special cases such as ALT_CTF in NSW), a value of 1 is assigned if the duration threshold is exceeded. If a CTF event spans multiple water-years, the value of 1 is applied only in the year the event finishes. This column aligns with original LTWP development and is recommended for annual reporting.

numAchieved: Number of EWR achievements recorded in the year. This reflects how many times the full EWR criteria were met within the year and is relevant for EWRs that allow or require multiple achievements per year.

numEvents: Number of successful EWR events in the year. Each event is counted when it exceeds the minimum required duration and meets all threshold and timing requirements. Values greater than one may occur for EWRs such as baseflows or very low flows when the duration requirement is met multiple times within the same year.

numEventsAll: Total number of detected events in the year that meet threshold and timing requirements, regardless of whether duration or other success criteria are met. This includes both successful and unsuccessful events.

eventLength: Average duration (days) of all detected events in the year that meet threshold and timing requirements, including unsuccessful events.

eventLengthAchieved: Average duration (days) of successful events only, where all EWR requirements (threshold, timing, and duration) are satisfied.

totalEventDays: Total number of days in the year where flow, level, or volume exceeds the EWR threshold within the required timing window. This includes days associated with unsuccessful events.

totalEventDaysAchieved: Total number of event days in the year associated only with successful events.

maxEventDays: Maximum duration (days) of any successful event occurring within the year.

maxRollingEvents: Maximum event duration (days) accumulated up to the water-year boundary. If an event crosses a water-year boundary, its total duration continues to accumulate into the next year. This metric is primarily used in CTF frequency calculations.

maxRollingAchievement: Indicates whether the rolling event duration meets or exceeds the minimum annual duration requirement for CTF EWRs. A value of 1 indicates compliance; 0 indicates non-compliance.

4.8.3 Inter-event statistics columns

rollingMaxInterEvent: Maximum inter-event gap (days) accumulated up to the water-year boundary. If an inter-event period crosses a water-year boundary, the total number of gap days continues to accumulate into the next year. This metric is calculated using the *all_successful_interEvents* table.

RollingMaxInterEventAchieved: Indicates whether the rolling maximum inter-event period for the year complies with the LTWP requirement. A value of 1 indicates the inter-event gap is less than or equal to the target maximum; 0 indicates the maximum inter-event period has been exceeded**.**

4.8.4 Interpretation notes

- eventYears is best suited for LTWP-aligned annual reporting
- numEventsAll and totalEventDays support diagnostic and sensitivity analyses
- rollingMaxInterEvent and RollingMaxInterEventAchieved are critical for understanding prolonged dry periods
- Data availability fields should always be reviewed alongside achievement metrics to contextualise results.

4.9 Events and inter-events comparison

The EWR Tool produces four event-based and inter-event-based outputs. These tables differ in whether duration requirements are applied and whether the output represents events or gaps between events. This structure allows users to undertake both standard compliance reporting and more detailed sensitivity or diagnostic analyses. **Error! Reference source not found.** shows event and inter-event comparison and purpose.

Table 1 Events and inter-events comparison

Output	Represents	Threshold & Timing Applied	Minimum Duration Applied	Purpose
all_events	Individual events	Yes	No	Captures all detected events that meet flow/level/volume thresholds within the defined timing window, regardless of duration.
all_interEvents	Gaps between events	Yes (via all_events)	No	Records inter-event periods between all detected events, including unsuccessful events.
all_successful_events	Individual events	Yes	Yes	Contains only events that meet the minimum duration requirements defined in the LTWPs/EWMPs.
all_successful_interevents	Gaps between successful events	Yes	Yes	Records inter-event periods only between successful EWR events.

In addition to event-based results, the tool also reports:

- inter-event periods between successful events, which are used to assess compliance with maximum inter-event requirements
- all inter-event gaps, including those between non-successful events, to support further investigation of flow regime behaviour.

The outputs distinguish between:

- successful EWR achievements, where all requirements for an EWR code are met
- multiple successful events within a single year, where applicable
- partial or non-successful events, where flow, level, or volume thresholds are met within the required timing window, but duration or frequency requirements are not satisfied.

4.10 All events results (all_events)

The all_events Pandas DataFrame records every individual hydrological event that meets an EWR’s timing window and flow, volume, or water level threshold, regardless of duration or minimum event requirements. Events listed in the results may or may not ultimately count as successful EWR events; duration and achievement checks occur later in the workflow (for example, in yearly_results and summary outputs). Each row represents one detected event.

4.10.1 General columns

Scenario, Gauge, PlanningUnit, State, SWSDLName, Code, Multigauge (same as above), waterYear (same as Year).

4.10.2 Event timing columns

startDate: Start date of the event (DD/MM/YYYY). This is the first day on which the flow, volume, or water level meets the EWR threshold within the allowed timing window.

endDate: End date of the event (DD/MM/YYYY). This is the final day associated with the event, including any allowable gaps where a within-event gap tolerance applies.

4.10.3 Event duration columns

These columns distinguish between hydrological continuity and threshold exceedance, which is particularly important for EWRs that allow short interruptions below the threshold.

eventDuration: Total duration of the event in days, including days where flow, volume, or water level temporarily falls below the threshold when a “within-event gap tolerance” is specified. This represents the full span of the event from start to end, accounting for allowable short gaps.

eventLength: Effective event length in days, counting only the days where the flow, volume, or water level meets or exceeds the EWR threshold. Days below the threshold within an allowable gap tolerance are excluded. This value will be less than or equal to eventDuration and differs only for EWRs that define a within-event gap tolerance.

4.10.4 Interpretation notes

- all_events does not apply minimum duration or minimum number of events
- Events listed here may later be classified as successful (counted in yearly_results.numEvents) or excluded from achievement metrics if duration requirements are not met
- The distinction between eventDuration and eventLength is critical for EWRs that permit short interruptions, such as ecological flow pulses with allowable dry breaks
- This table is a useful tool to understand why an EWR fails due to insufficient durations.

4.11 All inter-events results (all_interEvents)

The all_interEvents Pandas DataFrame represents the inter-event periods between all detected EWR events, regardless of whether those events meet minimum duration or achievement requirements. This result is the inverse of the all_events DataFrame and records the time gaps between consecutive events that meet the EWR timing and flow, volume, or water level threshold criteria, without considering duration or frequency requirements.

This DataFrame is useful for exploratory analysis and sensitivity testing where understanding all flow or level breaks between events is required.

4.11.1 General columns

Scenario, Gauge, PlanningUnit, State, SWSDLName, Code, Multigauge (same as above).

4.11.2 Event timing columns

startDate, endDate (same as above).

4.11.3 Inter-event duration column

InterEventLength: Length of the inter-event period in days, representing the number of consecutive days between EWR events.

4.11.4 Interpretation notes

- Each row represents a gap between two consecutive EWR events, regardless of whether those events were successful
- Inter-event periods in this table are not filtered by minimum duration or achievement rules
- This output provides a complete picture of event spacing and is particularly useful for understanding flow intermittency and supporting sensitivity or diagnostic analyses.

For LTWP compliance and formal inter-event assessment, users should refer to the all_successful_interevents and yearly_results outputs.

4.12 All successful events results (all_successful_events)

The all_successful_events Pandas DataFrame is a filtered subset of the all_events output. It retains only those events that meet the minimum duration requirement defined for each EWR in the LTWPs, in addition to satisfying the timing window and flow, volume, or water level threshold requirements. This DataFrame therefore represents events that are considered successful at the event level, prior to applying any annual aggregation rules such as required number of events per year or inter-event constraints.

4.12.1 General columns

Scenario, Gauge, PlanningUnit, State, SWSDLName, Code, Multigauge (same as above), waterYear (same as Year).

4.12.2 Event timing columns

startDate, endDate (same as above).

4.12.3 Event duration columns

eventDuration, eventLength (same as above).

4.12.4 Interpretation notes

- Every row in all_successful_events represents an event that meets minimum duration requirements, except specific very low flow and baseflow EWRs, where duration is assessed annually
- These results form the basis for: numEvents and numAchieved in yearly_results, achievement counts and frequencies in summary_results, inter-event gap calculations for successful events
- Events excluded from this table are still available in all_events for diagnostic and sensitivity analyses.

4.13 All successful inter-events results (all_successful_interevents)

The all_successful_interevents Pandas DataFrame represents the inter-event periods between successful EWR events. It is effectively the inverse of the all_successful_events DataFrame and records the time gaps between consecutive successful events for each EWR at a given site.

Only inter-event periods that occur between events that have met the minimum EWR duration requirements are included. These inter-events are used to assess compliance with maximum allowable inter-event periods specified in the LTWPs.

This DataFrame is a key input to: Rolling inter-event calculations in the yearly_results output, and determining whether maximum inter-event thresholds are exceeded for an EWR.

4.13.1 General columns

Scenario, Gauge, PlanningUnit, State, SWSDLName, Code, Multigauge (same as above).

4.13.2 Event timing columns

startDate, endDate (same as above).

4.13.3 Inter-event column

InterEventLength: Length of the inter-event period in days, representing the number of consecutive days between successful EWR events.

4.13.4 Interpretation notes

- Each row represents a gap between two successful EWR events
- Inter-event periods are assessed against the maximum inter-event duration targets defined in LTWPs
- Rolling inter-event durations derived from this table are used to populate: rollingMaxInterEvent and RollingMaxInterEventAchieved in the yearly_results output.

Inter-event periods between unsuccessful events, or between unsuccessful and successful events, are not included in this table and are instead captured in the all_interEvents output.

Appendix A

A.1 Input model data type descriptions

A.1.1 “Bigmod - MDBA” format

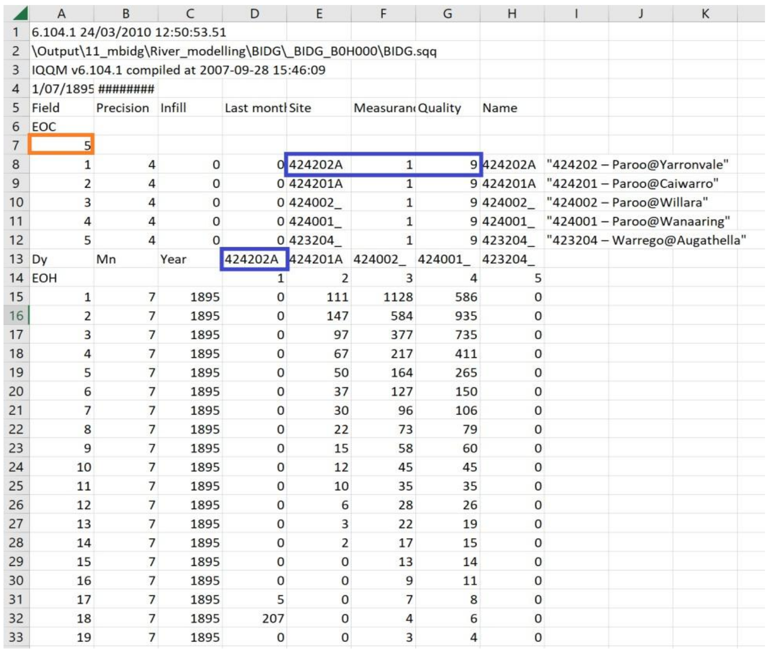
An example of the “Bigmod - MDBA” time-series format is shown in Figure 4 (colours are used for emphasis in this user guide only). The value highlighted in orange (cell A6) specifies the number of sites included in the dataset. This value must be accurate. If time series are added, removed, or otherwise modified, the site count must be updated accordingly.

The EWR Tool ingests:

- Header data (rows 1 to 8, inclusive), followed by
- Flow data (from row 9 onwards).

The header data is used to construct new column headings based on the site–measurand–quality information contained within the header. These constructed headings form the key metadata used by the tool in subsequent processing steps. Once generated, they replace the original column headings in the dataset.

Care must be taken when editing BigMod time-series files. If header rows are altered or deleted without making corresponding changes to the associated data columns, the generated header list may no longer align with the data. This mismatch can lead to incorrect ingestion and erroneous EWR assessment results.



	A	B	C	D	E	F	G	H	I	J	K
1	6.104.1 24/03/2010 12:50:53.51										
2	\Output\11_mbidg\River_modelling\BIDG_BIDG_B0H000\BIDG.sqq										
3	IQQM v6.104.1 compiled at 2007-09-28 15:46:09										
4	1/07/1895	########									
5	Field	Precision	Infill	Last montl	Site	Measuran	Quality	Name			
6	EOC										
7	5										
8	1	4	0	0	424202A	1	9	424202A	"424202 – Paroo@Yarronvale"		
9	2	4	0	0	424201A	1	9	424201A	"424201 – Paroo@Caiwarro"		
10	3	4	0	0	424002_	1	9	424002_	"424002 – Paroo@Willara"		
11	4	4	0	0	424001_	1	9	424001_	"424001 – Paroo@Wanaaring"		
12	5	4	0	0	423204_	1	9	423204_	"423204 – Warrego@Augathella"		
13	Dy	Mn	Year	424202A	424201A	424002_	424001_	423204_			
14	EOH			1	2	3	4	5			
15	1	7	1895	0	111	1128	586	0			
16	2	7	1895	0	147	584	935	0			
17	3	7	1895	0	97	377	735	0			
18	4	7	1895	0	67	217	411	0			
19	5	7	1895	0	50	164	265	0			
20	6	7	1895	0	37	127	150	0			
21	7	7	1895	0	30	96	106	0			
22	8	7	1895	0	22	73	79	0			
23	9	7	1895	0	15	58	60	0			
24	10	7	1895	0	12	45	45	0			
25	11	7	1895	0	10	35	35	0			
26	12	7	1895	0	6	28	26	0			
27	13	7	1895	0	3	22	19	0			
28	14	7	1895	0	2	17	15	0			
29	15	7	1895	0	0	13	14	0			
30	16	7	1895	0	0	9	11	0			
31	17	7	1895	5	0	7	8	0			
32	18	7	1895	207	0	4	6	0			
33	19	7	1895	0	0	3	4	0			

1. Figure 4 Example of BigMod format data (Source: **NSW long-term water plans environmental water requirement assessment code description 2024)**

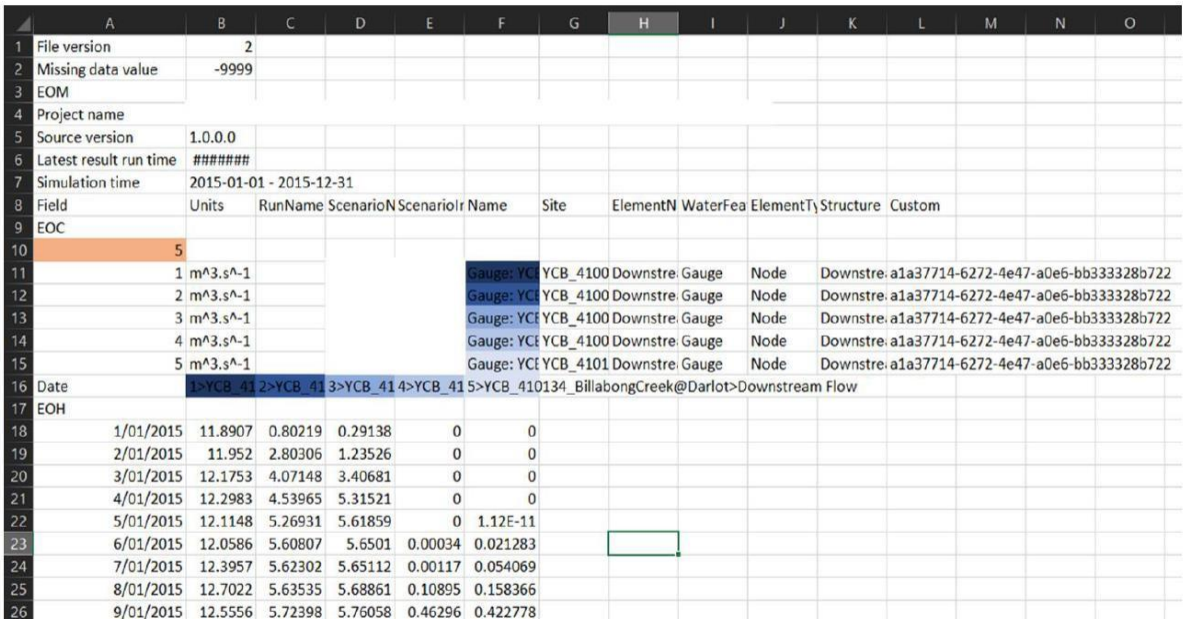
A.1.2 “Source - NSW (res.csv)” format

An example of a “Source - NSW (res.csv)” output file is shown in **Error! Reference source not found.** (highlighting is used for illustration purposes in this user guide only). A key element of this format is the **EOC** value, which in this example is 5 (shaded orange), indicating that the file contains data for five sites. This value must be consistent with the number of sites represented in the file. The EWR Tool reads the header information from the “Name” column and uses these entries to construct the flow data column headings.

The mapping follows the order of the data, such that:

- the top value in the “Name” column is assigned to the left-most flow data column
- subsequent values are assigned sequentially to the remaining flow data columns.

Once constructed, these flow data headings are matched to their corresponding gauges using the SiteID.csv file located in the model_metadata folder. The tool then checks the matched gauges against the EWR database to identify and assess any applicable EWRs for each location.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	File version	2													
2	Missing data value	-9999													
3	EOM														
4	Project name														
5	Source version	1.0.0.0													
6	Latest result run time	#######													
7	Simulation time	2015-01-01 - 2015-12-31													
8	Field	Units	RunName	ScenarioN	Scenariolr	Name	Site	ElementN	WaterFea	ElementTy	Structure	Custom			
9	EOC														
10	5														
11	1	m^3.s^-1				Gauge: YCE	YCB_4100	Downstre	Gauge	Node	Downstre	a1a37714-6272-4e47-a0e6-bb333328b722			
12	2	m^3.s^-1				Gauge: YCE	YCB_4100	Downstre	Gauge	Node	Downstre	a1a37714-6272-4e47-a0e6-bb333328b722			
13	3	m^3.s^-1				Gauge: YCE	YCB_4100	Downstre	Gauge	Node	Downstre	a1a37714-6272-4e47-a0e6-bb333328b722			
14	4	m^3.s^-1				Gauge: YCE	YCB_4100	Downstre	Gauge	Node	Downstre	a1a37714-6272-4e47-a0e6-bb333328b722			
15	5	m^3.s^-1				Gauge: YCE	YCB_4101	Downstre	Gauge	Node	Downstre	a1a37714-6272-4e47-a0e6-bb333328b722			
16	Date	1>YCB_41	2>YCB_41	3>YCB_41	4>YCB_41	5>YCB_410134_BillabongCreek@Darlot>Downstream Flow									
17	EOH														
18	1/01/2015	11.8907	0.80219	0.29138	0	0									
19	2/01/2015	11.952	2.80306	1.23526	0	0									
20	3/01/2015	12.1753	4.07148	3.40681	0	0									
21	4/01/2015	12.2983	4.53965	5.31521	0	0									
22	5/01/2015	12.1148	5.26931	5.61859	0	1.12E-11									
23	6/01/2015	12.0586	5.60807	5.6501	0.00034	0.021283									
24	7/01/2015	12.3957	5.62302	5.65112	0.00117	0.054069									
25	8/01/2015	12.7022	5.63535	5.68861	0.10895	0.158366									
26	9/01/2015	12.5556	5.72398	5.76058	0.46296	0.422778									

2. Figure 5 Example of Source format data (Source: **NSW long-term water plans environmental water requirement assessment code description 2024)**

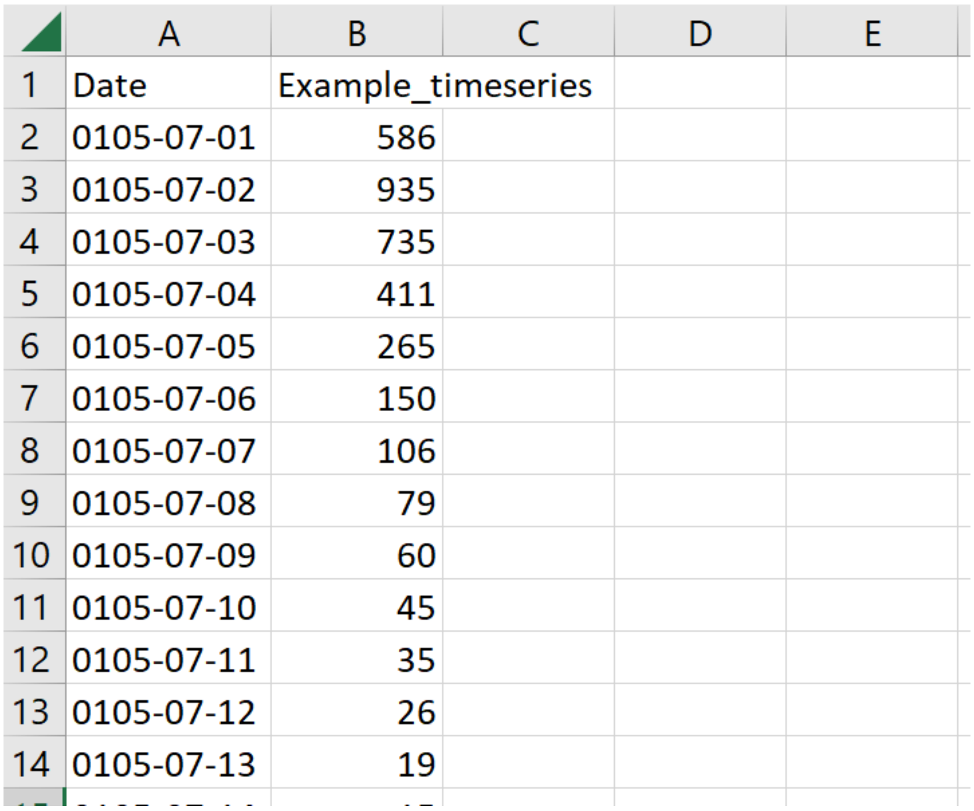
A.1.3 “ten thousand year” format

The EWR Tool has been designed to support 10,000-year climate time series, as required by New South Wales. An example of this file format is shown in **Error! Reference source not found.**.

This option requires input data in CSV (.csv) format and must include:

- a “Date” column, and
- at least one data column with a heading that contains a gauge identifier (highlighted in blue in Figure 4 for illustration purposes only).

Provided the gauge identifier is included in the column heading, the tool can detect it automatically and use it to link the time series to the appropriate EWRs in the database.

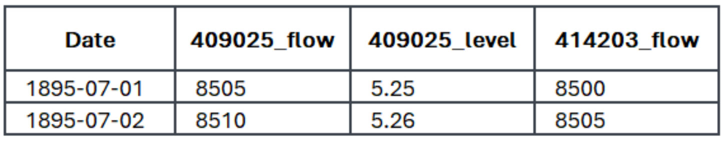


	A	B	C	D	E
1	Date	Example_timeseries			
2	0105-07-01	586			
3	0105-07-02	935			
4	0105-07-03	735			
5	0105-07-04	411			
6	0105-07-05	265			
7	0105-07-06	150			
8	0105-07-07	106			
9	0105-07-08	79			
10	0105-07-09	60			
11	0105-07-10	45			
12	0105-07-11	35			
13	0105-07-12	26			
14	0105-07-13	19			

3. Figure 6 Example of ten thousand years model data (Source: **NSW long-term water plans environmental water requirement assessment code description 2024)**

A.1.4 “Standard time-series" format

The standard time-series format don’t have meta data format and only contains “Date” column (YYYY-MM-DD) and gauges with either _flow or _level data as shown in Figure 7.



Date	409025_flow	409025_level	414203_flow
1895-07-01	8505	5.25	8500
1895-07-02	8510	5.26	8505

Figure 7 Example of Standard time-series model data

Appendix B

B.1 Common EWR parameters defined across Basin state governments

Below are the definitions of the some of the common parameters that make up the majority of EWRs across Basin state governments and how they relate to elements of the hydrograph. In parentheses are the corresponding column names in the parameter_sheet.csv that hold the information, and their units. For some EWRs, specific parameters are not covered here, or specific calculations will be required. These will be defined in future versions of this document.

B.1.1 Standard flow EWR parameters

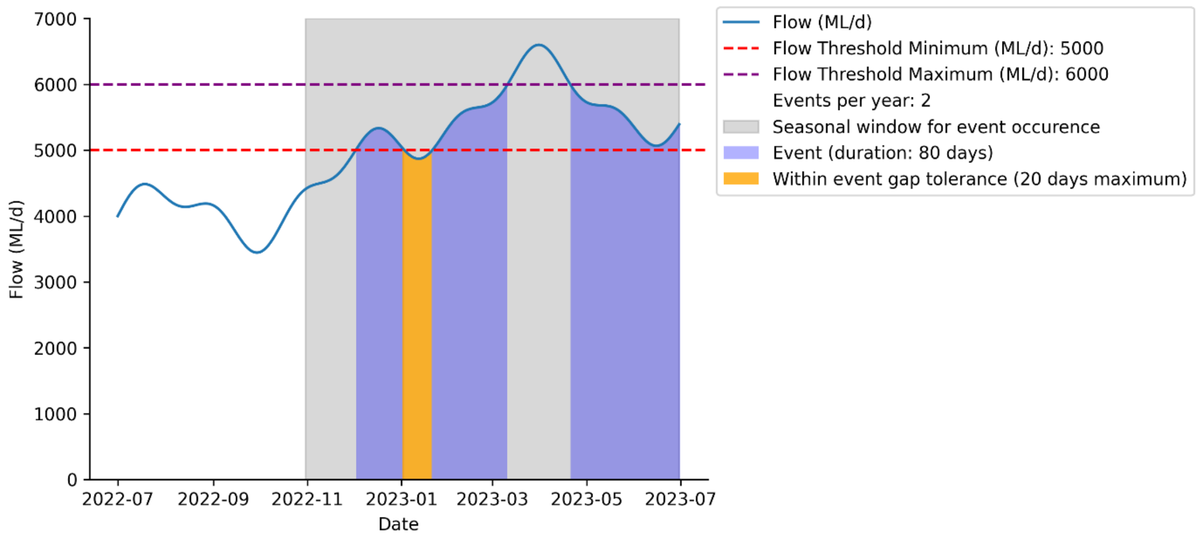


Figure 8 An example hydrograph showing some common flow parameters found across different EWRs, these include: start month and end month, flow threshold minimum and maximum, duration, and within event gap tolerance. Two events are recorded under this particular EWR's parameters.

Timing (StartMonth (month), EndMonth (month)): Expressed as a range of months of the year in which the flow event is required or preferred to occur. This is either any time of the water year (July – June) Days within the timing months are inclusive, meaning the window begins on day 1 of the start month and ends on the last day of the end month. For some EWRs, start or end month may be a decimal number e.g. 7.15. This indicates the seasonal window begins in the middle of the month starting on the 15th day. The timing window is seen as a hard edge in the assessment code and will end an EWR event.

Duration (Duration (days)): Events are consecutive days and the minimum duration specified in the LTWP and EWMPs.

Minimum flow threshold (FlowThresholdMin ML/d): Minimum flow that needs to be exceeded or equalled; for example, a flow greater than or equal to 10,000 ML/d (expressed as ≥10,000 ML/d).

Minimum spell (MinSpell (days)): Minimum individual spell length that can contribute towards the duration. It is usually either a 1 or equal to the duration. When it is a 1 this implies that the duration is a non-consecutive requirement. If the minimum spell and duration are the same, it means the duration is a consecutive requirement.

Within event gap tolerance (WithinEventGapTolerance (days)): Maximum number of consecutive days that the flow can fall below the flow threshold assigned to the EWR and still be considered a continuous event. The EWR has to start above the required flow threshold before it can drop below it.

Events per year (EventsPerYear (#)): The number of events occurring within the year for said year to be considered a successful event year for the EWR. For example: EWR A has an events per year number of 2. One 1 event is recorded in a given year, therefore this is not considered a successful event year.

B.1.2 Standard volumetric EWR parameters

In conjunction with standard parameters shared across EWRs assessing flow (ML/d), additional parameters related to cumulative volume can be defined.

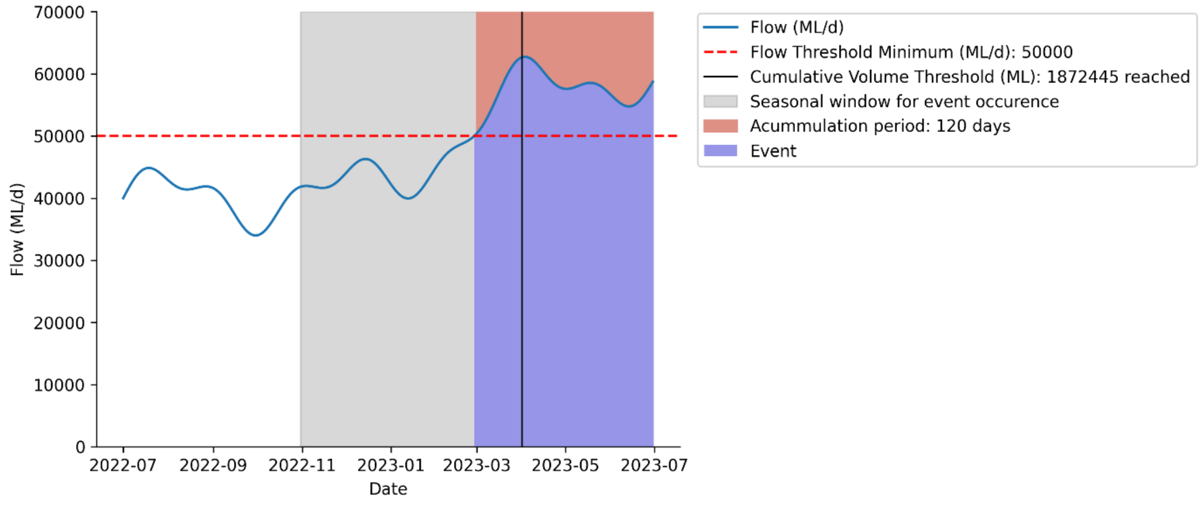


Figure 9 An example hydrograph showing some common volume parameters found across different EWRs, these include: start month and end month (seasonal window for event occurrence), flow threshold minimum, and accumulation period. One event is recorded where the cumulative volume is greater than or equal to the volume threshold.

Accumulation period (AccumulationPeriod (days)): The number of days in which a cumulative volume threshold is to be reached over the required accumulation period below (greater than or equal to).

Flow volume (Volume (ML)): The minimum cumulative volume to be reached over the required accumulation period below (greater than or equal to)

B.1.3 Standard level EWR parameters

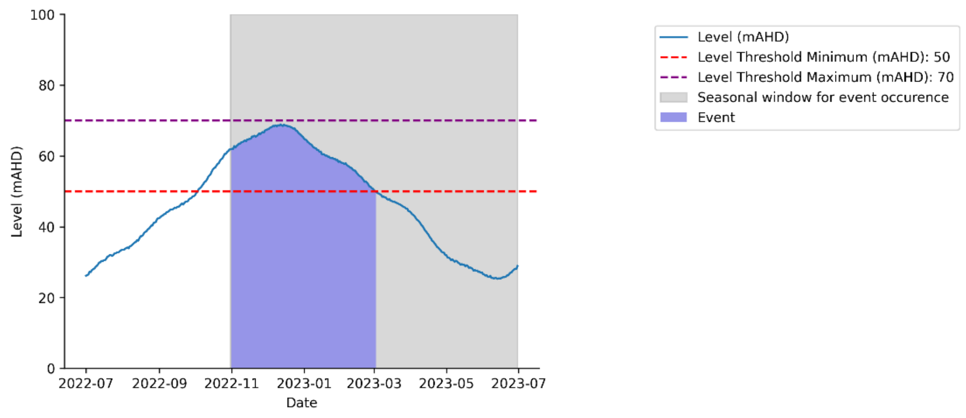


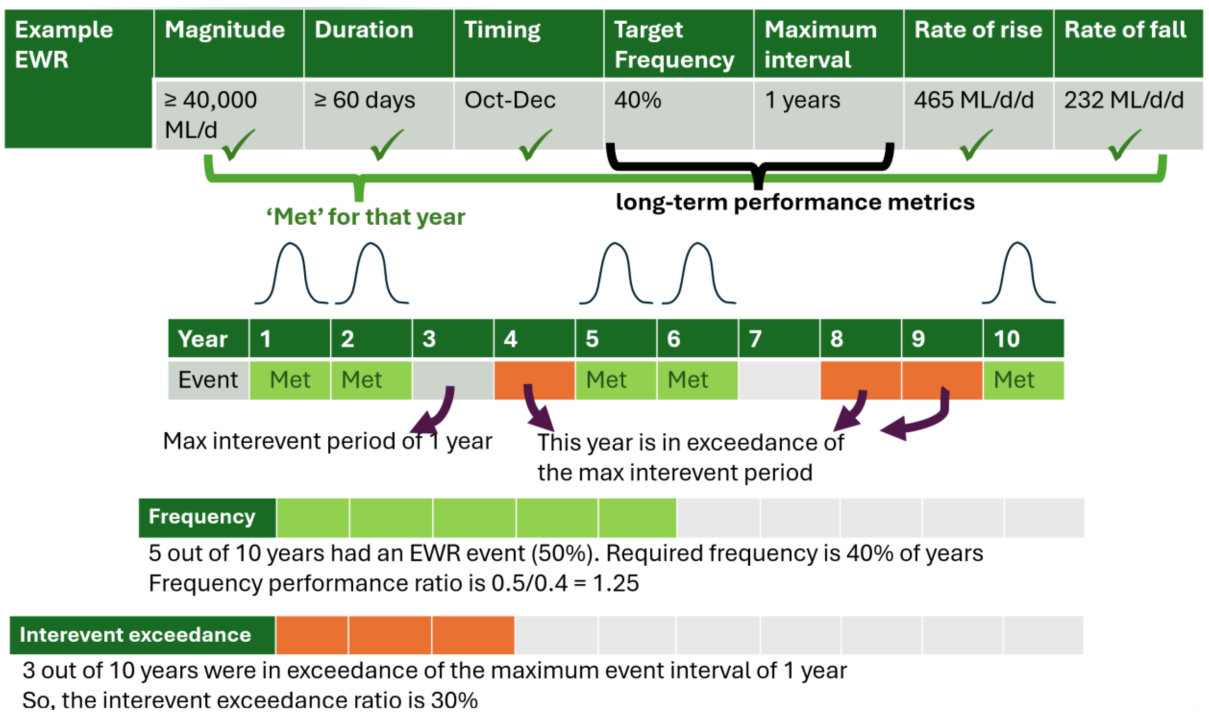
Figure 10 An example hydrograph showing common level parameters found across different EWRs, these include: start month and end month (seasonal window for event occurrence) and level threshold minimum and maximum.

B.2 Long term metrics applying to most EWRs

For most EWRs, long term performance is measured either with Target Frequency and or Maximum recommended Interevent Period which are also recorded in the parameter sheet. Which long term metrics are relevant to a given EWR is specified in their respective long term water plans and Environmental water management plans for each state. These long term metrics are not used in any calculations by the EWR tool but are included in the output to facilitate further analysis, in particular for comparing EWR frequency to the target frequency and EWR interevent lengths to the maximum recommended interevent period.

Target Frequency (TargetFrequency (%)): The target frequency of EWR events over the entire hydrograph period for example, for a target frequency of 50% and a hydrograph of 4 years length, 2 out of the 4 years needs an event achievement for the target frequency to be met.

Maximum recommended interevent period (MaxInter-event (years)): Expressed in years within the parameter sheet. It is the period between successful EWR events, when flow exceeds the minimum threshold and meets other parameter requirement such as seasonality and duration.



Example EWR	Magnitude	Duration	Timing	Target Frequency	Maximum interval	Rate of rise	Rate of fall
	≥ 40,000 ML/d ✓	≥ 60 days ✓	Oct-Dec ✓	40%	1 years	465 ML/d/d ✓	232 ML/d/d ✓

Year	1	2	3	4	5	6	7	8	9	10
Event	Met	Met		🟧	Met	Met		🟧	🟧	Met

'Met' for that year
long-term performance metrics
Max interevent period of 1 year
This year is in exceedance of the max interevent period
Frequency
Interevent exceedance

5 out of 10 years had an EWR event (50%). Required frequency is 40% of years
Frequency performance ratio is 0.5/0.4 = 1.25

3 out of 10 years were in exceedance of the maximum event interval of 1 year
So, the interevent exceedance ratio is 30%

Figure 9 Example of metrics for an EWR and how comparison with these long term metrics can be calculated to gauge long term performance of an EWR over the span of the entire hydrograph. Here, ratios are taken to measure the exceedance of interevent periods and target frequencies.

Office locations
Adelaide – *Kaurna*
Canberra – *Ngunnawal*
Goondiwindi – *Bigambul*
Griffith – *Wiradjuri*
Mildura – *Latji Latji*
Murray Bridge – *Ngarrindjeri*
Wodonga – *Dhudhuroa*

mdba.gov.au 1800 630 114 engagement@mdba.gov.au