Report Object Model - Class Reference

Complete class signatures for Report→UUT/UUR hierarchy

Last Updated: February 8, 2026

Core Report Hierarchy

WATSBase

Base: `object`

```
class WATSBase:
    # Properties
    id: str
    station_id: str
    root_part_num: str
    root_serial_num: str
    root_failed: bool
    created_datetime: datetime
    modified_datetime: datetime
    execution_time_sec: float

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> WATSBase
    __eq__(other) -> bool
    __repr__() -> str
```

Report

Base: `WATSBase`
Generic: `TypeVar('TSubUnit', bound=SubUnit)`

```
class Report(Generic[TSubUnit], WATSBase):
    # Properties
    process_code: str
    wats_version: str
    info: ReportInfo
    sub_units: List[TSubUnit]
    assets: List[Asset]
    binary_data: List[BinaryData]
    misc_info: MiscInfo
    additional_data: List[AdditionalData]

    # Methods
    __init__(...)
    get_asset(asset_id: str) -> Optional[Asset]
    get_binary_data(binary_id: str) -> Optional[BinaryData]
    get_subunit_by_id(subunit_id: str) -> Optional[TSubUnit]
    get_subunit_by_serial(serial: str) -> Optional[TSubUnit]
    to_dict() -> dict
    from_dict(data: dict) -> Report
```

UUTReport

Base: `Report<UUTSubUnit>`

```
class UUTReport(Report[UUTSubUnit]):
    # Inherited from Report<UUTSubUnit>
    info: UUTInfo  # Specialized type
    sub_units: List[UUTSubUnit]  # Specialized type

    # Additional Properties
    run: int

    # Additional Methods
    get_all_steps() -> List[Step]
    get_steps_by_type(step_type: Type[Step]) -> List[Step]
    get_failed_steps() -> List[Step]
    get_all_measurements() -> List[BaseMeasurement]
    get_failed_measurements() -> List[BaseMeasurement]
    find_step_by_path(path: str) -> Optional[Step]
    calculate_yield() -> float
```

UURReport

Base: `Report<UURSubUnit>`

```
class UURReport(Report[UURSubUnit]):
    # Inherited from Report<UURSubUnit>
    info: UURInfo  # Specialized type
    sub_units: List[UURSubUnit]  # Specialized type

    # Additional Properties
    process_code_repair: str
    process_code_test: str
    ref_uut: str  # Reference to original UUT report

    # Additional Methods
    get_all_failures() -> List[UURFailure]
    get_failures_by_component(component: str) -> List[UURFailure]
    get_subunit_hierarchy() -> Dict[str, List[UURSubUnit]]
    link_to_uut(uut_report: UUTReport) -> None
```

SubUnit Hierarchy

SubUnit

Base: `object`

```
class SubUnit:
    # Properties
    id: str
    serial_num: str
    part_num: str
    failed: bool
    test_socket_index: int
    batch_serial: str
    head_placement: int
    site_placement: int

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> SubUnit
    __eq__(other) -> bool
    __repr__() -> str
```

UUTSubUnit

Base: `SubUnit`

```
class UUTSubUnit(SubUnit):
    # Additional Properties
    steps: List[Step]

    # Additional Methods
    get_step_by_id(step_id: str) -> Optional[Step]
    get_root_sequence() -> Optional[SequenceCall]
    get_all_measurements() -> List[BaseMeasurement]
    calculate_test_time() -> float
```

UURSubUnit

Base: `SubUnit`

```
class UURSubUnit(SubUnit):
    # Additional Properties
    idx: int  # Index in hierarchy
    parent_idx: int  # Parent index (-1 for root)
    failures: List[UURFailure]

    # Additional Methods
    get_parent(all_subunits: List[UURSubUnit]) -> Optional[UURSubUnit]
    get_children(all_subunits: List[UURSubUnit]) -> List[UURSubUnit]
    is_root() -> bool
    add_failure(failure: UURFailure) -> None
```

Info Classes

ReportInfo

Base: `object`

```
class ReportInfo:
    # Properties
    operator_name: str
    workorder: str
    comments: str
    started_datetime: datetime
    finished_datetime: datetime
    hostname: str

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> ReportInfo
```

UUTInfo

Base: `ReportInfo`

```
class UUTInfo(ReportInfo):
    # Additional Properties
    test_program_name: str
    test_program_version: str
    revision_num: str
    fixture_id: str

    # No additional methods
```

UURInfo

Base: `ReportInfo`

```
class UURInfo(ReportInfo):
    # Additional Properties
    defect_code: str
    repair_action: str
    repair_location: str
    repair_type: str
    technician_id: str
    rework_count: int

    # No additional methods
```

Step Hierarchy (UUT Reports)

Step

Base: `object`

```
class Step:
    # Properties
    id: str
    name: str
    type: str
    result: str  # "Passed", "Failed", "Skipped", etc.
    started_datetime: datetime
    execution_time_sec: float
    parent_id: Optional[str]

    # Methods
    __init__(...)
    is_passed() -> bool
    is_failed() -> bool
    to_dict() -> dict
    from_dict(data: dict) -> Step
```

SequenceCall

Base: `Step`

```
class SequenceCall(Step):
    # Additional Properties
    children: List[Step]
    info: SequenceCallInfo
    loop_info: Optional[LoopInfo]

    # Additional Methods
    add_child(step: Step) -> None
    get_child_by_id(step_id: str) -> Optional[Step]
    get_all_descendants() -> List[Step]
    get_failed_children() -> List[Step]
    get_depth() -> int
```

NumericStep

Base: `Step`

```
class NumericStep(Step):
    # Additional Properties
    measurement: NumericMeasurement

    # Additional Methods
    get_value() -> float
    is_within_limits() -> bool
```

MultiNumericStep

Base: `Step`

```
class MultiNumericStep(Step):
    # Additional Properties
    measurements: List[NumericMeasurement]

    # Additional Methods
    get_values() -> List[float]
    get_failed_measurements() -> List[NumericMeasurement]
```

PassFailStep

Base: `Step`

```
class PassFailStep(Step):
    # Additional Properties
    measurement: BooleanMeasurement

    # Additional Methods
    get_value() -> bool
```

StringStep

Base: `Step`

```
class StringStep(Step):
    # Additional Properties
    measurement: StringMeasurement

    # Additional Methods
    get_value() -> str
```

MultipleStringStep

Base: `Step`

```
class MultipleStringStep(Step):
    # Additional Properties
    measurements: List[StringMeasurement]

    # Additional Methods
    get_values() -> List[str]
```

GenericStep

Base: `Step`

```
class GenericStep(Step):
    # Additional Properties
    data: dict

    # No additional methods
```

ActionStep

Base: `Step`

```
class ActionStep(Step):
    # Additional Properties
    action_type: str
    parameters: dict

    # No additional methods
```

ChartStep

Base: `Step`

```
class ChartStep(Step):
    # Additional Properties
    chart_data: List[float]
    x_axis_label: str
    y_axis_label: str

    # Additional Methods
    get_chart_data() -> List[float]
```

UnknownStep

Base: `Step`

```
class UnknownStep(Step):
    # Additional Properties
    raw_data: dict

    # No additional methods
```

Measurement Classes (UUT Reports)

BaseMeasurement

Base: `object`

```
class BaseMeasurement:
    # Properties
    id: str
    name: str
    status: str
    comp_op: CompOp

    # Methods
    __init__(...)
    is_passed() -> bool
    is_failed() -> bool
    to_dict() -> dict
    from_dict(data: dict) -> BaseMeasurement
```

NumericMeasurement

Base: `BaseMeasurement`

```
class NumericMeasurement(BaseMeasurement):
    # Additional Properties
    value: float
    low_limit: Optional[float]
    high_limit: Optional[float]
    units: str

    # Additional Methods
    is_within_limits() -> bool
    get_deviation() -> float
    get_percentage_of_range() -> float
```

BooleanMeasurement

Base: `BaseMeasurement`

```
class BooleanMeasurement(BaseMeasurement):
    # Additional Properties
    value: bool
    expected: bool

    # Additional Methods
    matches_expected() -> bool
```

StringMeasurement

Base: `BaseMeasurement`

```
class StringMeasurement(BaseMeasurement):
    # Additional Properties
    value: str
    expected: Optional[str]

    # Additional Methods
    matches_expected() -> bool
```

CompOp (Enum)

```
class CompOp(Enum):
    EQ = "EQ"      # Equal
    NE = "NE"      # Not Equal
    LT = "LT"      # Less Than
    LE = "LE"      # Less or Equal
    GT = "GT"      # Greater Than
    GE = "GE"      # Greater or Equal
    GELE = "GELE"  # Between (inclusive)
    GELT = "GELT"  # Between (exclusive high)
    GTLT = "GTLT"  # Between (exclusive)
```

UUR-Specific Classes

UURFailure

Base: `object`

```
class UURFailure:
    # Properties
    id: str
    component_name: str
    component_designator: str
    failure_code: str
    failure_description: str
    symptom: str
    root_cause: str
    corrective_action: str
    defect_found: bool
    defect_induced: bool

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> UURFailure
```

UURMiscInfo

Base: `MiscInfo`

```
class UURMiscInfo(MiscInfo):
    # Additional Properties
    repair_duration_min: int
    parts_replaced: List[str]
    verification_test_passed: bool

    # Additional Methods
    add_replaced_part(part: str) -> None
```

Supporting Classes

Asset

Base: `object`

```
class Asset:
    # Properties
    id: str
    name: str
    asset_type: str
    file_path: str
    mime_type: str
    size_bytes: int

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> Asset
    is_image() -> bool
    is_pdf() -> bool
```

BinaryData

Base: `object`

```
class BinaryData:
    # Properties
    id: str
    name: str
    data: bytes
    mime_type: str

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> BinaryData
    to_base64() -> str
    from_base64(encoded: str) -> BinaryData
```

Attachment

Base: `object`

```
class Attachment:
    # Properties
    id: str
    name: str
    file_path: str
    description: str

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> Attachment
```

AdditionalData

Base: `object`

```
class AdditionalData:
    # Properties
    key: str
    value: Any
    data_type: str

    # Methods
    __init__(...)
    to_dict() -> dict
    from_dict(data: dict) -> AdditionalData
```

MiscInfo

Base: `object`

```
class MiscInfo:
    # Properties
    custom_fields: dict

    # Methods
    __init__(...)
    add_field(key: str, value: Any) -> None
    get_field(key: str) -> Optional[Any]
    to_dict() -> dict
    from_dict(data: dict) -> MiscInfo
```

SequenceCallInfo

Base: `object`

```
class SequenceCallInfo:
    # Properties
    call_index: int
    recursion_depth: int

    # Methods
    __init__(...)
    to_dict() -> dict
```

LoopInfo

Base: `object`

```
class LoopInfo:
    # Properties
    current_iteration: int
    total_iterations: int

    # Methods
    __init__(...)
    to_dict() -> dict
```

Quick Reference

UUT Report Structure

```
UUTReport
├── info: UUTInfo
├── sub_units: List[UUTSubUnit]
│   └── steps: List[Step]
│       ├── SequenceCall (contains children)
│       ├── NumericStep (has NumericMeasurement)
│       ├── MultiNumericStep (has List[NumericMeasurement])
│       ├── PassFailStep (has BooleanMeasurement)
│       ├── StringStep (has StringMeasurement)
│       ├── MultipleStringStep (has List[StringMeasurement])
│       ├── GenericStep
│       ├── ActionStep
│       ├── ChartStep
│       └── UnknownStep
├── assets: List[Asset]
├── binary_data: List[BinaryData]
├── misc_info: MiscInfo
└── additional_data: List[AdditionalData]
```

UUR Report Structure

```
UURReport
├── info: UURInfo (includes repair data)
├── process_code_repair: str
├── process_code_test: str
├── ref_uut: str (link to original UUT)
├── sub_units: List[UURSubUnit]
│   ├── idx: int (hierarchy index)
│   ├── parent_idx: int (parent reference)
│   └── failures: List[UURFailure]
├── assets: List[Asset]
├── binary_data: List[BinaryData]
├── misc_info: UURMiscInfo (extends with repair fields)
└── additional_data: List[AdditionalData]
```

Class Count Summary

Core Hierarchy: 7 classes

- WATSBase, Report, UUTReport, UURReport, SubUnit, UUTSubUnit, UURSubUnit

Info Classes: 3 classes

- ReportInfo, UUTInfo, UURInfo

Step Classes: 11 classes

- Step, SequenceCall, NumericStep, MultiNumericStep, PassFailStep, StringStep, MultipleStringStep, GenericStep, ActionStep, ChartStep, UnknownStep

Measurement Classes: 4 classes + 1 enum

- BaseMeasurement, NumericMeasurement, BooleanMeasurement, StringMeasurement, CompOp

UUR-Specific: 2 classes

- UURFailure, UURMiscInfo

Supporting Classes: 7 classes

- Asset, BinaryData, Attachment, AdditionalData, MiscInfo, SequenceCallInfo, LoopInfo

Total: 34 classes + 1 enum

Usage Examples

Creating a UUT Report

```python
from pywats import PyWATS

wats = PyWATS(host="localhost", port=8086)

# Get report
report = wats.report.get_uut_report("REPORT_ID")

# Access hierarchy
print(f"Process: {report.process_code}")
print(f"Run: {report.run}")
print(f"Serial: {report.root_serial_num}")

# Access UUT-specific info
print(f"Test Program: {report.info.test_program_name}")
print(f"Operator: {report.info.operator_name}")

# Navigate sub-units and steps
for subunit in report.sub_units:
    print(f"SubUnit: {subunit.serial_num}")
    for step in subunit.steps:
        if isinstance(step, NumericStep):
            print(f"  {step.name}: {step.measurement.value} {step.measurement.units}")
```

Creating a UUR Report

```python
# Get repair report
uur = wats.report.get_uur_report("UUR_ID")

# Access UUR-specific info
print(f"Repair Process: {uur.process_code_repair}")
print(f"Test Process: {uur.process_code_test}")
print(f"Defect Code: {uur.info.defect_code}")
print(f"Repair Action: {uur.info.repair_action}")

# Navigate hierarchy
hierarchy = uur.get_subunit_hierarchy()
for parent_id, children in hierarchy.items():
    print(f"Parent: {parent_id}")
    for child in children:
        print(f"  Child: {child.serial_num} (idx={child.idx})")
        for failure in child.failures:
            print(f"    Failure: {failure.failure_code} - {failure.component_name}")
```

See Also:

- UUT Object Model Diagram
- UUR Object Model Diagram
- Complete API Reference