Objectives

This document explains what each major method and class in the Objectives utility does, using plain, non-programmer language. It is intended for physicists, dosimetrists, and developers who want to understand the behavior of the code without reading the source.

CopyObjectivesFromPlanToPlan

This method copies all optimization objectives from one treatment plan to another.

What it does:
- Checks that both plans exist
- Confirms both plans belong to the same patient
- Confirms both plans use the same structure set (This check is disabled for OART)
- Reads every optimization objective from the source plan
- Recreates those objectives on the target plan

When you would use it:
- Creating a new plan that should behave like an existing plan
- Duplicating optimization logic without manually re-entering objectives

Important limitations:
- Line objectives are not supported and will be skipped
- If any single objective fails, the copy process stops

RemoveAllOptimizationObjectives

This method deletes every optimization objective from a plan.

What it does:
- Confirms the plan and optimization exist
- Collects all current objectives
- Removes them one by one

When you would use it:
- Resetting a plan before rebuilding objectives
- Cleaning up corrupted or imported optimization data

Safety behavior:
- Individual removal failures are logged (commented out)
- The method attempts to continue removing remaining objectives

RemoveOptimizationObjectivesByStructureId

This method removes only the optimization objectives associated with one structure.

What it does:
- Finds objectives tied to the specified structure
- Removes only those objectives

When you would use it:
- Reworking objectives for a specific target or OAR
- Fixing duplicated objectives on a single structure

GetTargetsByLowerObjective

This method identifies structures that have a lower (minimum dose) optimization objective.

What it does:
- Scans optimizer objectives
- Finds targets constrained by minimum dose requirements

When you would use it:
- Identifying targets

GetOARbyNoLowerObjective

This method identifies structures that do NOT have a lower dose objective.

What it does:
- Reviews optimizer objectives
- Finds OARs without minimum dose constraints

When you would use it:
- Quality assurance checks
- Identifying missing protection objectives

DynamicObjective (Class)

DynamicObjective is a portable representation of an optimization objective.

Why it exists:
- ESAPI optimization objectives cannot be cloned directly
- This class allows objectives to be stored, copied, compared, and recreated

What it supports:
- Point objectives
- Mean dose objectives
- EUD objectives

What it does NOT support:
- Line objectives

DynamicObjective.AddObjective

This method adds a stored objective back into a treatment plan.

What it does:
- Finds the structure in the plan
- Adds the correct objective type
- Validates dose, volume, and priority values

Safety behavior:
- Invalid volumes are automatically corrected to 0-100 (e.g. cannot be negative or >100).
- Unsupported objective types are skipped

Line Objective Conversion

Line objectives describe an entire DVH curve, but they cannot be directly copied.

What this code does:
- Samples DVH curve points
- Keeps a configurable percentage of points
- Converts them into point objectives

Why this is useful:
- Line objectives cannot be copied or modified, but points can be.

ConvertAllLineObjectivesToPoints

This method converts every line objective in a plan into point objectives.

GetPointRepresentationOfLineObjective

This method shows what point objectives WOULD be created from a line objective, without modifying the plan.

What it does:
- Reads DVH curve data
- Calculates representative dose-volume points

Structures

AddNewStructureWithRandomId

Purpose:
Creates a new structure with a guaranteed unique ID.

Actual behavior:

- Generates a random, non-existing structure ID
- Calls ESAPI AddStructure
- Optionally converts the structure to high resolution
- Returns the new structure if successful

Error handling:
- Any exception is caught
- The method returns null
- The exception is swallowed (no logging or rethrow)

Notes:
Failure is silent. Callers must explicitly check for null.

CropStructure

Purpose:
Subtracts expanded OAR volumes from a base structure.

Actual behavior:
- Missing crop structures are skipped
- Temporary structures are created for margin expansion
- High-resolution mismatches are automatically resolved
- Boolean subtraction is performed on the base structure
- Temporary structures are cleaned up

Failure behavior:
- Skipped structures do not cause failure
- Boolean subtraction failure causes the method to return false

CropSIBTargets

Purpose:
Creates SIB-specific cropped target structures based on dose hierarchy.

Actual behavior:
- Existing structures may be replaced or renamed
- Structures are copied, colored, and cropped
- Higher-dose targets are subtracted from lower-dose targets

Failure behavior:
- One target failing does not stop others
- Failed targets are omitted from the result list

Notes:
The result list may be incomplete if failures occur.

EnsureHighResolution

Purpose:
Guarantees structures are usable for Boolean operations.

Actual behavior:
- High-resolution structures are reused
- Low-resolution structures are cloned into temporary high-resolution structures
- SegmentVolume, color, and comments are preserved
- A crosswalk maps original IDs to temporary IDs

Notes:
Temporary structures must be cleaned up after use.

RemoveStructures

Purpose:
Subtracts multiple structures from a base structure.

Actual behavior:
- Missing structures are ignored
- Resolution is enforced before Boolean operations
- Crosswalk mapping ensures correct geometry
- Temporary structures are removed at the end

Failure behavior:
- Any exception returns false
- Partial subtraction may already have occurred

Notes:
This method is not transactional.

ESAPI_Helpers

Purpose:
Provides utility functions missing from ESAPI.

Key behaviors:
- LightenColor safely clamps RGB values
- GenerateRandomString ensures valid ID generation
- Dose unit is derived from the plan prescription

Important note:
Dose unit retrieval fails if DosePerFraction is not defined.

RetrieveItems

Purpose:
Provides safe, case-insensitive retrieval of ESAPI objects.

Actual behavior:
- Duplicate matches throw explicit errors
- Missing items return null or empty collections
- Prevents subtle ESAPI lookup bugs

Notes:
Improves reliability of automation scripts.

Safety Summary

Overall characteristics:
- Resilient by design, favors continuity over strict failure
- Silent skips are intentional and common
- Boolean geometry operations are guarded

Key takeaway:
Always validate outputs when using these utilities in automation.

RapidPlan Utilities

RapidPlan.Add.Apply (Plan-only overload)

Purpose:
Applies DVH estimates to a plan using a specified RapidPlan model.

What actually happens:
- Verifies required inputs are present
- Retrieves the dose unit from the plan prescription
- Confirms the plan contains at least one treatment beam
- Builds dictionaries for target dose levels and structure matches
- Calls ESAPI CalculateDVHEstimates

Failure behavior:
- Missing prescription causes a hard exception
- Missing treatment beams causes a clean false return
- DVH calculation failure throws an exception

Notes:
This method is not silent. Many failures will throw and must be handled by the caller.

RapidPlan.Add.Apply (ESAPI validation overload)

Purpose:
Applies DVH estimates after validating inputs against the ESAPI environment.

What actually happens:
- Validates structure existence in the plan
- Confirms RapidPlan model exists
- Verifies model structure compatibility
- Delegates execution to the core Apply method

Failure behavior:
- Validation failures return false
- Runtime failures throw exceptions

RapidPlan.Add.Apply (Serialized setup overload)

Purpose:
Applies a RapidPlan configuration stored in serialized form.

What actually happens:
- Translates serialized objects into runtime tuples
- Skips invalid or incomplete entries
- Reuses existing Apply logic

Failure behavior:
- Invalid entries are silently skipped

CopyRapidPlanFromSourceToTargetPlan

Purpose:
Copies RapidPlan DVH estimation settings from one plan to another.

What actually happens:
- Reads calculation logs from the source plan
- Extracts RapidPlan model ID and structure mappings
- Does NOT rely on DVH estimate objects
- Applies the extracted configuration to the target plan

Failure behavior:
- Missing model ID causes clean failure
- Missing structure mappings causes clean failure
- DVH estimation failures propagate as exceptions

Notes:
This method assumes the source plan already successfully ran RapidPlan.

RapidPlan.Validation.ValidateRapidPlanInputs

Purpose:
Ensures a RapidPlan configuration is safe to apply.

What actually happens:
- Confirms plan has a structure set
- Confirms all referenced structures exist in the plan
- Confirms the RapidPlan model exists in Eclipse
- Confirms structure IDs exist in the model

Failure behavior:
- Validation issues return false
- Unexpected ESAPI failures throw exceptions

Notes:
This method prevents applying incompatible RapidPlan configurations.

RapidPlan.Retrieval.Extract

Purpose:
Extracts RapidPlan configuration from an existing plan.

What actually happens:
- Verifies the plan contains treatment beams
- Verifies DVH Estimation logs exist
- Parses logs only (no DVH objects used)
- Builds a serialized RapidPlan configuration

Failure behavior:
- Invalid plans return an empty configuration
- Parsing errors throw exceptions

Notes:
Extraction depends on log integrity.

GetRapidPlanModelIdFromLogs

Purpose:
Finds the RapidPlan model ID used for DVH estimation.

What actually happens:
- Searches DVH Estimation calculation logs
- Uses regex to extract model name

Failure behavior:
- Returns null if no model ID is found

Notes:
This method cannot infer models that were never run.