# Pascalスクリプト APIリファレンス

このドキュメントは、Munition_AutoPatcher_v1.1 プロジェクトで使用される主要なPascalスクリプトの構造と、各関数の役割を解説します。

---

## 処理の呼び出し階層

xEditからのスクリプト実行は、以下の階層で行われます。

1. **`00_RunAllExtractors.pas`**: xEditから直接呼び出されるエントリーポイント。
2. **`AutoPatcherCore.pas`**: 各抽出処理の実行関数（`AP_Run_...`）を呼び出すインターフェース。
3. **`*Logic.pas`** (`ExtractWeaponAmmoMappingLogic.pas` など): 各抽出処理の具体的な実装。
4. **`AutoPatcherLib.pas`**: すべてのスクリプトから利用される汎用ヘルパー関数ライブラリ。

---

### 1. `00_RunAllExtractors.pas` (統合実行スクリプト)

このスクリプトは、xEditから一度だけ呼び出され、必要な全てのデータ抽出処理を順番に実行する「司令塔」の役割を担います。

| 関数名 | 役割 | 戻り値 |
|---|---|---|
| `Initialize` | スクリプトのエントリーポイント。`AutoPatcherCore.pas` の `AP_Run_...` 関数を順番に呼び出します。 | `Integer` (0: 全て成功, 1: いずれかが失敗) |

---

### 2. `AutoPatcherCore.pas` (コアロジック)

各抽出処理の実行関数（インターフェース）を定義し、具体的な処理を各`*Logic.pas`ファイルに委譲します。

| 関数名 | 役割 | 戻り値 |
|---|---|---|
| `AP_Run_ExtractWeaponAmmoMapping` | 武器と弾薬のマッピング情報を抽出します。(`ExtractWeaponAmmoMappingLogic.pas` を呼び出します) | `Integer` (0: 成功, 1: 失敗) |
| `AP_Run_ExportWeaponLeveledLists` | 武器関連のレベルドリストを抽出します。(`ExportLeveledListsLogic.pas` を呼び出します) | `Integer` (0: 成功, 1: 失敗) |
| `AP_Run_ExportMunitionsAmmoIDs` | `Munitions - An Ammo Expansion.esl` から弾薬のFormIDとEditorIDを抽出し、`munitions_ammo_ids.ini` を生成します。 | `Integer` (0: 成功, 1: 失敗) |
| `AP_Run_GenerateStrategyFromMunitions` | Munitionsの弾薬情報を元に `strategy.json` の雛形を生成します。(**注:** 現在はPython側の `run_strategy_generation` でより高度な処理が行われているため、この関数はレガシー機能です) | `Integer` (0: 成功, 1: 失敗) |
| `AP_Run_ExportWeaponAmmoDetails` | 武器とそれが使用する弾薬の詳細情報を抽出し、`weapon_ammo_details.txt` を生成します。`robco_ini_generate.py` で利用されます。 | `Integer` (0: 成功, 1: 失敗) |

---

### 3. `ExtractWeaponAmmoMappingLogic.pas` (武器・弾薬マッピング)

武器（WEAP）レコードを走査し、使用弾薬とのマッピング情報を抽出する具体的なロジックを実装しています。

| 関数名 | 役割 | 戻り値 |
|---|---|---|
| `AP_Run_ExtractWeaponAmmoMapping` | 全てのプラグインを走査し、武器とその弾薬の関連情報を抽出します。以下の2つのファイルを生成します。<br>- `weapon_ammo_map.json`: 武器のEditorIDと弾薬のFormIDのマップ。<br>- `unique_ammo_for_mapping.ini`: 抽出された弾薬の一覧。 | `Integer` (0: 成功, 1: 失敗) |

---

### 4. `ExportLeveledListsLogic.pas` (レベルドリスト抽出)

武器が配布される可能性のあるレベルドリスト（LVLI）を抽出し、CSVファイルとして出力するロジックを実装しています。

| 関数名 | 役割 | 戻り値 |
|---|---|---|
| `AP_Run_ExportWeaponLeveledLists` | 全てのプラグインを走査し、武器関連のキーワードを含むレベルドリストを抽出します。`WeaponLeveledLists_Export.csv` を生成します。 | `Integer` (0: 成功, 1: 失敗) |

### 5. `AutoPatcherLib.pas` (汎用ライブラリ)

複数のスクリプトから共通して利用される、汎用的なヘルパー関数群を提供します。

| 関数名 | 役割 |
|---|---|
| `GetOutputDirectory` | xEditの標準出力ディレクトリパス (`...\Edit Scripts\Output\`) を取得します。 |
| `LogSuccess`, `LogError`, etc. | `[AutoPatcher] SUCCESS:` のような接頭辞付きで、統一されたフォーマットのログを出力します。 |
| `SaveJSONToFile`, `SaveINIToFile` | `TStringList` の内容をファイルに保存し、成功/失敗ログを自動で出力します。 |
| `CreateMasterExclusionList` | `Fallout4.esm` やDLCなど、処理対象外とするマスターファイルのリストを生成します。 |
| `GetFullFormID` | レコードのロードオーダーを考慮した完全なFormID (例: `FE008001`) を取得します。 |
| `LowerTrim`, `SplitCSVToList`, etc. | 文字列の整形やリスト操作を行うための汎用ユーティリティです。 |
| `GetEditorIdSafe` | `EditorID()` で取得できない場合でも、`GetElementEditValues` を使って安全にEditorIDを取得します。 |

### 6. 生成されるファイルと役割

xEditスクリプトは、後続のPython処理で利用される中間ファイルを `Output` ディレクトリに生成します。

| ファイル名 | 生成元スクリプト | 内容 | 利用先 (Python) |
|---|---|---|---|
| `weapon_ammo_map.json` | `ExtractWeaponAmmoMappingLogic.pas` | 武器（WEAP）のEditorIDと、それが使用する弾薬（AMMO）のFormIDを関連付けたJSON配列。 | `robco_ini_generate.py`: 武器と弾薬の基本的な関連を把握するために使用。 |
| `unique_ammo_for_mapping.ini` | `ExtractWeaponAmmoMappingLogic.pas` | `weapon_ammo_map.json` の中で見つかった、Munitions以外のMODが追加したユニークな弾薬のリスト。`[UnmappedAmmo]`セクションに `FormID=ESP名\|EditorID` の形式で記録される。 | `mapper.py`: このリストを元に、ユーザーが手動でMunitions弾薬への変換（マッピング）を行うための入力データとして使用。 |
| `WeaponLeveledLists_Export.csv` | `ExportLeveledListsLogic.pas` | 武器が配布される可能性のあるレベルドリスト（LVLI）の情報をCSV形式で出力したもの。`EditorID`, `FormID`, `SourceFile`などの列を含む。 | `robco_ini_generate.py`: `LLI_Hostile_Gunner_Any` などの特定のレベルドリストのFormIDを解決するために使用。これにより、Robco Patcherがどのレベルドリストに武器を追加すべきかを判断できる。 |

| ファイル名 | 生成元スクリプト | 内容 | 利用先 (Python) |
| --- | --- | --- | --- |
| munitions_ammo_ids.ini | AutoPatcherCore.pas (AP_Run_ExportMunitionsAmmoIDs) | Munitions - An Ammo Expansion.esl に含まれる全ての弾薬のFormIDとEditorIDをINI形式で出力したもの。[MunitionsAmmo]セクションに FormID=EditorID の形式で記録される。 | Orchestrator.py (run_strategy_generation): このリストと分類ルール (ammo_categories.json) を照合し、各弾薬のカテゴリ（Pistol, Rifleなど）を決定する。 mapper.py: ユーザーが弾薬をマッピングする際の、変換先候補リストとして使用。 |
| weapon_ammo_details.txt | AutoPatcherCore.pas (AP_Run_ExportWeaponAmmoDetails) | MODが追加した武器と、それが使用する弾薬の詳細情報（プラグイン名、FormID、EditorID）を \| 区切りで出力したもの。 | robco_ini_generate.py: Robco Patcherがレベルドリストに武器を追加する際の、武器の完全な識別情報（Plugin\|FormID）を取得するために使用。 |