PUBLISHING HANDOVER DOCUMENT

# SDC·MDIS 통합포털서비스 퍼블리싱 개발 전달 문서

Header · Footer 템플릿 코드 교체 요청
모달 팝업 · 파일 업로드 컴포넌트 사용 가이드

| | |
|---|---|
| 프로젝트 | **SDC·MDIS 통합포털서비스** |
| 문서 버전 | **v1.0** |
| 작성일 | **2026. 08. 13.** |
| 작성 | **퍼블리싱팀** |
| 수신 | **개발팀** |

본 문서는 퍼블리싱 산출물의 변경 사항과 신규 컴포넌트의 적용 방법을 개발팀에 전달하기 위한 문서입니다.
문서에 포함된 코드는 산출물 원본에서 발췌한 것으로, 그대로 복사하여 사용할 수 있습니다.

# 목차


## 1. Header / Footer 구조 변경 안내 — 템플릿 코드 교체 요청

## 2. 모달 팝업 사용 방법

2.1 동작 구조 요약

2.2 기본 마크업 및 클래스 규칙

2.3 모달 크기(폭) 조정 방법

2.4 modal-footer 고정형 사용법

2.5 접근성 처리 (자동 적용 범위)

2.6 개발 시 주의사항

## 3. 파일 업로드 컴포넌트 사용 방법

3.1 구성 요소 및 클래스 역할

3.2 일반형 (체크박스 없음) — 화면 예시 포함

3.3 개별관리형 file-each-manage (체크박스 생성) — 화면 예시 포함

3.4 has-file 클래스 자동 토글 동작

3.5 JS가 생성하는 목록 DOM 구조

3.6 삭제 버튼 동작 분기

3.7 개발 시 주의사항

# 1. Header / Footer 구조 변경 안내

> **Header 및 Footer 구조가 전면 변경되었습니다 — 템플릿 코드 전체 교체를 요청드립니다.**
>
> 기존 공통 템플릿(include 파일)의 Header · Footer 마크업을
> **신규 산출물의 해당 영역으로 통째로 교체**
> 해 주십시오. 클래스명·계층 구조가 모두 달라졌으므로
> **부분 수정으로는 신규 기능이 동작하지 않습니다.**

## 교체 대상 및 범위

전달 산출물 `html/layout_main.html` 을 기준으로 아래 **행 번호 구간을 그대로 복사**하여 기존 템플릿의 Header · Footer 영역과 교체해 주시기 바랍니다.

| 교체 영역 | layout_main.html 행 번호 | 시작 / 종료 코드 | 비고 |
|---|---|---|---|
| **Header 영역**<br>전자정부 배너 포함 | **40행 ~ 391행** | 시작 : `<div class="gov-eg">`<br>종료 : `<!-- //header -->` | `.gov-eg` (전자정부 배너)는 `<header>` **바깥**에 위치하므로 **반드시 함께 교체**해야 합니다. |
| **Footer 영역** | **695행 ~ 775행** | 시작 : `<!-- footer -->`<br>종료 : `<!-- //footer -->` | 관련사이트 모달이 Footer 내부에 포함되어 있으므로 구간 전체를 교체합니다. |

# 2. 모달 팝업 사용 방법

모달 팝업은 `js/common.js` 154~240행의 공통 로직으로 동작합니다. **별도 초기화 함수 호출이 필요 없으며**, 정해진 클래스 규칙에 맞춰 마크업만 작성하면 열기·닫기·포커스 트랩·ESC 닫기·배경 스크롤 잠금이 모두 자동 적용됩니다.

> **반드시 지켜야 할 3가지 규칙**
>
> ① 호출 버튼의 **data-name** 값과 모달의 **id** 값이 **정확히 일치**해야 합니다.
> ② 닫기 버튼은 반드시 **class="modal-close"** 로 처리해야 합니다.
> ③ 모달 폭 조정은 **.modal-body** 에 **style="max-width:○○px"** 로 처리합니다.

## 2.1 동작 구조 요약

| 단계 | 처리 내용 |
|---|---|
| `.modal-open` 클릭 | 버튼의 **data-name** 값과 **동일한 id** 를 가진 `.wrap-modal` 을 탐색 |
| 열기 | `.wrap-modal` 에 `.is-current` 부여 → `display:flex` 로 중앙 정렬 + 딤 배경 표시<br>`.modal-body` 에 `tabindex="0"` , `.scroll-y` 부여<br>`<html>` 에 `overflow:hidden` 적용 (배경 스크롤 잠금) |
| 포커스 | 모달 내 첫 포커스 요소로 자동 이동(10ms 지연) + Tab / Shift+Tab 순환(focus trap) |
| 닫기 | **.modal-close** 클릭 또는 **ESC 키**. 닫힌 후 **호출 버튼으로 포커스 복귀** |

## 2.2 기본 마크업 및 클래스 규칙

```
<!-- 1) 호출 버튼 : data-name = 모달의 id -->
<a href="javascript:void(0)" class="button modal-open" data-name="modal-data-info">행정통계자료정보관리</a>


<!-- 2) 모달 본체 : id = 호출 버튼의 data-name -->
<div class="wrap-modal" id="modal-data-info">
  <div class="modal-body">
    <h4 class="modal-tit">행정통계자료정보관리</h4>

    <div class="modal-cont">
      <!-- 본문 콘텐츠 -->
    </div>

    <button type="button" class="modal-close"><i>닫기</i></button>
  </div>
</div>
```

**필수 확인**

위 예시에서 **data-name="modal-data-info"** 와 **id="modal-data-info"** 는 **반드시 같은 값**이어야 합니다. 값이 다르면 **모달이 열리지 않으며**

, 닫은 후 호출 버튼으로 포커스가 복귀하지 않습니다.
닫기 버튼은 **class="modal-close"** 가 없으면
**닫기 동작은 물론 ESC 키 닫기까지 동작하지 않습니다.**

| 클래스 / 속성 | 필수 | 역할 |
|---|---|---|
| `.modal-open` | **필수** | 호출 버튼. **data-name** 속성 반드시 함께 사용 |
| **data-name** | **필수** | 열고자 하는 모달의 **id** 값과 동일하게 작성 |
| `.wrap-modal` + **id** | **필수** | 모달 최상위 래퍼. 기본 `display:none`, 열림 시 `.is-current` 부여 |
| `.modal-body` | **필수** | 실제 팝업 박스. 기본 `max-width:680px` (2.3절 폭 조정 참조) |
| **.modal-close** | **필수** | 닫기 버튼. **이 클래스가 없으면 닫기 · ESC 모두 동작하지 않음** |
| `.modal-tit` | 권장 | 팝업 제목. `strong` 또는 `h4` 등 사용 |
| `.modal-cont` | 권장 | 본문 영역. 내용이 길면 자동 스크롤 |
| `.modal-footer` | 선택 | 하단 고정 버튼 영역 (2.4절 참조) |

## 2.3 모달 크기(폭) 조정 방법

`.modal-body` 의 기본 폭은 `max-width:680px` 입니다. 이보다 넓은 팝업이 필요한 경우 **.modal-body 에 인라인 style 로 max-width 를 지정**합니다. 별도 클래스를 추가하는 방식이 아니므로 주의해 주십시오.

```
<div class="wrap-modal" id="modal-data-info">
  <div class="modal-body" style="max-width:1034px">      <!-- 폭 확장 -->
    <h4 class="modal-tit">행정통계자료정보관리</h4>
    <div class="modal-cont"> ... </div>
    <button type="button" class="modal-close"><i>닫기</i></button>
  </div>
</div>
```

**모달 사이즈 조정 규칙**

모달의 폭을 조정하려면 **반드시 .modal-body 요소에 style="max-width:○○px" 를 직접 지정**해 주십시오.
`.wrap-modal` 이나 `.modal-cont` 에 지정하면 적용되지 않습니다.

※ 높이는 `max-height:calc(100vh - 40px)` 로 자동 제한되며, 내용이 길어지면 `.modal-cont` 가 스크롤 처리되므로 별도 지정이 필요 없습니다.
※ 폭은 `width:calc(100% - 40px)` 기준의 반응형이므로, 지정한 `max-width` 는 상한값으로만 동작합니다.

## 2.4 modal-footer 고정형 사용법

`.modal-footer` 가 존재하면 CSS `:has()` 셀렉터에 의해 **하단 버튼 영역이 고정되고 본문( `.modal-cont` )만 스크롤**됩니다. (common.css 1029~1037행) 별도 스크립트나 추가 클래스는 필요 없습니다.

```
<div class="wrap-modal" id="modal-data-info">
  <div class="modal-body" style="max-width:1034px">
    <h4 class="modal-tit">행정통계자료정보관리</h4>

    <div class="modal-cont">
      <!-- 이 영역만 스크롤됨 -->
    </div>

    <!-- 하단 고정 영역 -->
    <div class="modal-footer board-button">
      <button type="button" class="button modal-close"><i>닫기</i></button>
    </div>

    <!-- 우측 상단 X 닫기 버튼 -->
    <a href="#" class="modal-close">닫기</a>
  </div>
</div>
```

**닫기 버튼은 2개까지 사용 가능**

우측 상단 X 버튼과 하단
**닫기**

버튼 모두 **.modal-close** 를 사용하며 개수 제한이 없습니다. CSS는 `[class*='close']:not([class*='button'])` 조건으로
**클래스에 `button` 이 포함되지 않은 요소만**

X 아이콘 형태로 렌더링합니다. 즉 하단 버튼에 `class="button modal-close"` 를 주면 일반 버튼 모양, `class="modal-close"` 만 주면 X 아이콘이 됩니다.

## 2.5 접근성 처리 (자동 적용 범위)

| 항목 | 처리 주체 | 내용 |
|---|---|---|
| 포커스 이동 | 자동 | 열림 시 모달 내 첫 포커스 요소로 이동 |
| 포커스 트랩 | 자동 | Tab / Shift+Tab 시 모달 내부에서만 순환 |
| ESC 닫기 | 자동 | **.modal-close** 클릭 이벤트를 트리거하는 방식 |
| 포커스 복귀 | 자동 | 닫기 시 **[data-name=모달id]** 버튼으로 복귀 |
| 배경 스크롤 잠금 | 자동 | `<html>` 에 `overflow:hidden` |
| 빈 포커스 앵커 | 자동 | `.layer-hide` 요소를 동적 삽입 후 닫을 때 제거 |

## 2.6 개발 시 주의사항

| No | 내용 |
|---|---|
| 1 | **data-name 값과 모달 id 는 반드시 일치**해야 합니다. 불일치 시 열리지 않으며, 닫은 후 포커스 복귀도 동작하지 않습니다. |
| 2 | 모달 **id** 는 **페이지 내 유일**해야 합니다. 목록에서 행마다 모달을 생성하는 경우 `modal-detail-1`, `modal-detail-2` 형태로 인덱스를 부여해 주세요. |
| 3 | 닫기 버튼은 **class="modal-close"** 로 처리해 주세요. 다른 클래스명을 사용하면 닫기 · ESC 모두 동작하지 않습니다. |
| 4 | 모달 폭 조정은 **.modal-body 의 style="max-width:◯◯px"** 로 처리합니다 (2.3절). |
| 5 | 이벤트가 `$(function(){...})` 내부에서 **`.click()` 직접 바인딩** 방식입니다. **Ajax로 모달을 동적 삽입하면 이벤트가 붙지 않습니다.** 동적 생성이 필요하면 `$(document).on('click','.modal-open',...)` 형태의 위임 방식으로 변경하거나, 삽입 후 재바인딩해 주세요. |
| 6 | 모달 내부에 `<form>` 을 배치할 경우 버튼에 `type="button"` 을 명시해 의도치 않은 submit을 방지해 주세요. |
| 7 | 모달을 **여러 개 동시에 열지 마십시오.** 닫기 로직이 `$('.modal-body').hasClass('scroll-y')` 로 전역 판단하므로, 중첩 시 배경 스크롤 잠금이 해제되지 않을 수 있습니다. |
| 8 | 모달 본문이 길어질 경우 `.modal-cont` 를 반드시 사용해 주세요. `.modal-body` 직속에 콘텐츠를 넣으면 `overflow: hidden` 으로 잘립니다. |

# 3. 파일 업로드 컴포넌트 사용 방법

파일 업로드는 `js/utill.js` 568~695행에서 처리됩니다. 드래그앤드롭 영역과 목록 영역이 **형제 관계**로 짝을 이루며, `file-each-manage` 클래스 유무에 따라 **목록 항목의 체크박스 생성 여부**가 결정됩니다. `has-file` 은 **JS가 자동으로 부여/제거하는 상태 클래스**로, 마크업에 직접 작성하지 않습니다.

## 3.1 구성 요소 및 클래스 역할

| 클래스 / 속성 | 작성 주체 | 역할 |
|---|---|---|
| `data-upload-form` | 퍼블/개발 | 드래그앤드롭 영역 식별용 속성. **이 속성이 있어야 업로드 기능이 초기화**됩니다 |
| `.wrap-upload-box` | 퍼블/개발 | 업로드 박스 스타일. 드래그 중에는 `.dragover` 자동 부여 |
| `.file-input` | 퍼블/개발 | `input[type=file]`. 화면에서 숨김 처리되며 `label` 로 조작 |
| `.file-list` | 퍼블/개발 | 업로드 목록 영역. **`.wrap-upload-box` 의 바로 다음 형제여야 함** |
| `.file-each-manage` | 퍼블/개발 | **이 클래스를 함께 주면 목록 항목에 체크박스가 생성됨** (3.3절) |
| `.file-list-info` | 퍼블/개발 | 등록 건수 + 일괄 버튼 영역. 기본 `display:none` |
| `.list-count b` | 퍼블/개발 | 등록 파일 개수가 자동 기입되는 위치. **내용은 비워둠** |
| `.list-module` | 퍼블/개발 | 삭제·다운로드 등 일괄 처리 버튼 그룹 |
| `.has-file` | **JS 자동** | 파일 1개 이상일 때 `.file-list` 에 자동 부여 → `.file-list-info` 노출 |
| `ul.down-item` | **JS 자동** | 파일 목록 `ul`. 파일 추가 시 동적 생성, 0개가 되면 삭제됨 |

## 3.2 일반형 — 체크박스 없음

목록에 파일명만 표시되며, 일괄 버튼은 전체 삭제 용도로 사용합니다. (사용 예: ppt-48.html 533행, ppt-61.html, ppt-82.html, ppt-143.html 등)

```
<!-- 파일 업로드 영역 박스 -->
<div class="wrap-upload-box" data-upload-form>
  <p>첨부할 파일을 여기에 끌어다 놓거나. 파일 선택 버튼을 직접 선택해주세요.</p>

  <label for="fileInput" class="button-fill01">파일선택</label>
  <input type="file" id="fileInput" class="file-input" multiple>
</div>

<!-- 업로드한 파일 목록영역 : 반드시 위 박스의 바로 다음 형제 -->
<div class="file-list">
  <div class="file-list-info">
    <div class="list-count">등록파일 : <b></b>개</div>
    <div class="list-module">
      <button type="button" class="button-size01-frame01 ico-delete01-after"><i>전체삭제</i></button>
    </div>
  </div>
</div>
```

**[화면 예시 ①] 초기 상태 — 파일 0개 / `.file-list` ( `has-file` 없음)**

첨부할 파일을 여기에 끌어다 놓거나. 파일 선택 버튼을 직접 선택해주세요.

파일선택

.file-list-info 영역 숨김 ( `display:none` ) — 등록 건수·버튼 미노출

**[화면 예시 ②] 파일 3개 업로드 후 — `.file-list` .has-file 자동 부여**

첨부할 파일을 여기에 끌어다 놓거나. 파일 선택 버튼을 직접 선택해주세요.

파일선택

**등록파일 : 3 개** 전체삭제

| | | | |
|---|---|---|---|
| 연구계획서.hwp [hwp, 245.3KB] | 다운로드 | 바로보기 | 삭제 |
| 활용자료목록.xlsx [xlsx, 88.1KB] | 다운로드 | 바로보기 | 삭제 |
| 사전협의결과.pdf [pdf, 1.2MB] | 다운로드 | 바로보기 | 삭제 |

↑ `ul.down-item` 및 항목 `li` 는 JS가 자동 생성 · 건수는 `.list-count b` 에 자동 기입

## 3.3 개별관리형 — `file-each-manage` (체크박스 생성)

`.file-list` 에 `.file-each-manage` 를 함께 주면, JS가 목록 항목을 생성할 때 파일명을 **`<span class="form-checkbox"><input type="checkbox"><label>` 구조로 감싸 체크박스를 함께 출력**합니다. 이를 통해 선택 항목만 삭제하거나 다운로드하는 개별 관리가 가능해집니다. (사용 예: ppt-108-110.html 522행 · 694행)

```
<!-- 파일 업로드 영역 박스 -->
<div class="wrap-upload-box" data-upload-form>
  <p>첨부할 파일을 여기에 끌어다 놓거나. 파일 선택 버튼을 직접 선택해주세요.</p>

  <label for="fileInput01" class="button-fill01">파일선택</label>
  <input type="file" id="fileInput01" class="file-input" multiple>
</div>

<!-- 업로드한 파일 목록영역 : file-each-manage 추가 → 체크박스 생성 -->
<div class="file-list file-each-manage">
  <div class="file-list-info">
    <div class="list-count">등록파일 : <b></b>개</div>
    <div class="list-module">
      <button type="button" class="button-size01 ico-delete01-after"><i>체크 항목 삭제</i></button>
      <button type="button" class="button-size01 ico-down01-after"><i>선택 서약서 다운로드</i></button>
    </div>
  </div>
</div>
```

**[화면 예시 ③] 개별관리형 — `.file-list` .file-each-manage .has-file / 목록에 체크박스 생성**


↑ 파일명 앞 **체크박스**가 `.file-each-manage` 에 의해 자동 생성 · 체크된 항목만 일괄 삭제/다운로드 대상

| 구분 | 일반형 `.file-list` | 개별관리형 `.file-list.file-each-manage` |
|---|---|---|
| **목록 항목 형태** | `<p>파일명 [확장자, 용량]</p>` | `<p><span class="form-checkbox">…</span></p>` **체크박스 생성** |
| **일괄 삭제 버튼 동작** | 전체 삭제 | 체크된 항목만 삭제 (체크 없으면 전체 삭제) |

| 구분 | 일반형 `.file-list` | 개별관리형 `.file-list.file-each-manage` |
|---|---|---|
| **선택 다운로드** | 해당 없음 | 체크박스 기준으로 개발단 구현 가능 |
| **주 사용처** | 연구계획서 등 단순 첨부 | 공동연구자 서약서 등 **파일별 개별 처리**가 필요한 경우 |

## 3.4 has-file 클래스 자동 토글 동작

`has-file` 은 **마크업에 작성하지 않습니다.** 아래 공통 함수가 파일 개수에 따라 자동 처리합니다.

```
// js/utill.js — updateFileState()

function updateFileState($container){
    const $fileList = $container.find('ul.down-item');
    const count = $fileList.find('li').length;

    // 등록 건수 자동 기입 : .list-count b
    $container.find('.list-count b').text(count);

    // 파일 1개 이상 → has-file 추가 / 0개 → 제거 + ul 삭제
    if (count > 0) {
        $container.addClass('has-file');
    } else {
        $container.removeClass('has-file');
        $fileList.remove();
    }
}
```

| 파일 개수 | `.file-list` 클래스 | 화면 표시 |
|---|---|---|
| 0개 (초기) | `file-list` | `.file-list-info` 가 `display:none` → **건수·버튼 영역 숨김** (화면 예시 ①) |
| 1개 이상 | `file-list has-file` | `.file-list-info` 가 `display:flex` → **건수·버튼 영역 노출**<br>`ul.down-item` 자동 생성 및 목록 표시 (화면 예시 ②③) |

**CSS 정의 위치**

`css/util.css` 450~453행

`.wrap-application .file-list .file-list-info { display:none; }`

`.wrap-application .file-list.has-file .file-list-info { display:flex; ... }`

**주의:**

셀렉터에 `.wrap-application` 이 포함되어 있으므로, 업로드 컴포넌트는 **`.wrap-application` 하위에 배치**되어야 스타일이 정상 적용됩니다. 신청서 외 영역에서 사용할 경우 CSS 셀렉터 확장이 필요합니다.

## 3.5 JS가 생성하는 목록 DOM 구조

파일 첨부 시 `.file-list` 내부에 아래 구조가 자동 생성됩니다. 개발단에서 **서버에 이미 저장된 파일을 초기 렌더링**할 때는 동일한 구조로 출력해 주시면 삭제·카운트 로직이 그대로 동작합니다.

### 일반형 생성 결과 (화면 예시 ②)

```
<ul class="down-item">
  <li>
    <p>연구계획서.hwp [hwp, 245.3KB]</p>
    <a href="javascript:void(0)" class="ico-down"><i>다운로드</i></a>
    <a href="javascript:void(0)" class="ico-view"><i>바로보기</i></a>
    <a href="javascript:void(0)" class="ico-del"><i>삭제</i></a>
  </li>
</ul>
```

### 개별관리형( `file-each-manage` ) 생성 결과 (화면 예시 ③)

```
<ul class="down-item">
  <li>
    <p>
      <span class="form-checkbox">
        <input type="checkbox" id="sel-file-1">
        <label for="sel-file-1"><i>보안서약서_홍길동.pdf [pdf, 128.4KB]</i></label>
      </span>
    </p>
    <a href="javascript:void(0)" class="ico-down"><i>다운로드</i></a>
    <a href="javascript:void(0)" class="ico-view"><i>바로보기</i></a>
    <a href="javascript:void(0)" class="ico-del"><i>삭제</i></a>
  </li>
</ul>
```

파일명 표기 형식은 `파일명 [확장자, 용량]` 이며, 용량은 `formatSize()` 함수가 B / KB / MB / GB 단위로 소수점 1자리까지 변환합니다. 체크박스 `id` 는 `sel-file-1` 부터 페이지 내 전역 일련번호로 부여됩니다.

## 3.6 삭제 버튼 동작 분기

| 버튼 | 셀렉터 | 동작 |
|---|---|---|
| 항목별 삭제 | `.down-item .ico-del` | 해당 `li` 만 제거 후 카운트·`has-file` 갱신 |
| 일괄 삭제 | `.list-module [class*="ico-delete01"]` | **체크된 항목이 있으면 → 체크된 항목만 삭제**<br>**체크된 항목이 없으면 → 전체 삭제**<br>일반형은 체크박스가 없으므로 항상 전체 삭제로 동작합니다 |

두 삭제 이벤트 모두 `$(document).on()` **위임 방식**으로 바인딩되어 있어, 동적으로 추가된 항목에도 정상 동작합니다.

## 3.7 개발 시 주의사항

| No | 내용 |
|---|---|
| 1 | `.file-list` 는 `.wrap-upload-box` **의 바로 다음 형제여야 합니다.** 스크립트가 `$dropArea.next('.file-list')` 로 탐색하므로, 사이에 다른 요소가 들어가면 동작하지 않습니다. |
| 2 | `has-file` 을 마크업에 직접 작성하지 마십시오. JS가 관리하는 상태 클래스입니다. |

| No | 내용 |
|---|---|
| 3 | `.list-count b` 내부는 **비워두십시오.** 서버에서 값을 넣어도 초기화 시 0으로 덮어써집니다. 저장된 파일이 있는 경우 `ul.down-item` 자체를 렌더링하면 개수가 자동 계산됩니다. |
| 4 | 한 페이지에 업로드 영역이 여러 개여도 **input `id` 는 JS가 `file-input-1`, `file-input-2` … 로 재부여**하고 `label [for]` 도 함께 갱신합니다. 다만 초기 마크업 단계에서도 중복되지 않게 작성해 주세요. |
| 5 | 현재 스크립트에는 **파일 확장자·용량·개수 제한 검증이 없습니다.** `appendFiles()` 함수에 서버 정책에 맞는 검증 로직을 추가해 주세요. |
| 6 | 화면 목록은 DOM 기준으로만 관리되며 **실제 `input.files` 와 동기화되지 않습니다.** ( `change` 이벤트 후 `$(this).val('')` 로 초기화) 실제 전송은 `FormData` 를 직접 구성하는 방식으로 개발단에서 구현해 주세요. |
| 7 | `.ico-down` / `.ico-view` 링크는 퍼블리싱 단계에서 `javascript:void(0)` 더미 상태입니다. 실제 다운로드·미리보기 URL 연결이 필요합니다. |
| 8 | 업로드 컴포넌트는 `.wrap-application` 하위에 배치해야 CSS가 적용됩니다 (3.4절 참조). |

> **문의**
> 본 문서 내용 중 마크업·CSS 관련 문의사항은 퍼블리싱팀으로 회신해 주시기 바랍니다. 마크업 수정이 필요한 사항은 확인 후 퍼블리싱 산출물에 반영하여 재전달해 드립니다.