Global Callback 가이드

Last update: 2025/01/10

목차

Global Callback 가이드

글로벌 콜백은 아래와 같은 기능을 제공한다.

구분	설명
목적	각 화면에서 발생하는 이벤트를 전역적으로 처리하기 위한 공통 기능을 글로벌 모듈에서 개발하기 위한 기능을 제공한다.
용어 정의	“콜백”은 특정 이벤트에 개발자가 지정한 이벤트 처리용 함수를 의미한다.
콜백 지정	해당 실행 파리미터에 콜백 함수 이름을 지정한다. 예: CALLBACK_FOCUSMOVE_KEY: “SYSUtil.CallbackFocusMoveKey”
콜백 리턴값	콜백 리턴값에 따라 이벤트에 대한 기본 동작 수행 여부를 제어할 수 있다.
일반 글로벌 콜백	일반 글로벌 콜백은 실행 파라미터에 지정한 경우 발생시 호출되는 콜백이다.
이벤트 글로벌 콜백	이벤트 글로벌 콜백은 실행 파라미터에 지정한 xFrame5 이벤트에 대해서 호출되는 콜백이다.
샘플 소스 및 형식	“화면”→”템플릿 화면” 팝업 창에서 “/HTML5/COMMON_MODULE/global_callback(글로벌 콜백)”을 선택하면 샘플소스를 확인할 수 있다.

템플릿 위치: /HTML5/COMMON_MODULE/global_callback

템플릿 파일

- global_callback.xml
- global_callback.js
- 새창으로 실행

일반 글로벌 콜백

일반 글로벌 콜백은 특정한 이벤트 발생시 호출되는 콜백으로 지정가능한 내용은 아래와 같다. (참고: 목록은 알파벳 순이다.)

상세 내용은 아래 실행 파라미터별 샘플 소스내 주석을 참조한다.

구분	내용
CALLBACK_BROWSER_DOCKING	전용브라우저가 런타임브라우저와 탭모드로 연동한 경우 탭모드에서 도킹상태가 변경되었을때 호출된다.
CALLBACK_EXCEL_DOWNLOAD	엑셀 다운로드 진행시, 다른 방식으로 수행하기 위한 콜백이다.
CALLBACK_EXCEL_UPLOAD	엑셀 업로드 진행시, 다른 방식으로 수행하기 위한 콜백이다.
CALLBACK_FOCUSMOVE_KEY	키 입력에 의해서 포커스 이동전에 호출되며 콜백 리턴값을 기준으로 포커스 이동을 제어한다.
CALLBACK_META_DATA	메타 데이터를 이용하여 컴포넌트 생성시 호출되는 콜백이다.
CALLBACK_META_LOAD	메타 데이터 로드시 호출되는 콜백이다.
CALLBACK_GRID_DOWNLOADEXCEL	그리드 컨텍스트 메뉴의 “엑셀 저장” 또는 그리드 downloadexcel API 호출시 호출된다.

구분	내용
CALLBACK_GRID_DRMFILEREAD	전용 브라우저 환경경서 DRM_APPLY_TYPE 실행 파라미터가 2인 경우, 그리드 DRM 파일 읽기 전/후 호출된다.
CALLBACK_GRID_EXCEL_LOAD	그리드에 엑셀 업로드 완료 후 호출된다.
CALLBACK_GRID_PREKEYDOWN	그리드 키 처리 전에 호출되며, 리턴값을 기준으로 그리드 키 처리 여부를 결정된다.
CALLBACK_GRID_UPLOADEXCEL	그리드 컨텍스트 메뉴의 " 엑셀 불러오기 " 또는 그리드 uploadexcel API 호출시 호출된다.
CALLBACK_GRID_SAVECSV	그리드 컨텍스트 메뉴의 "CSV 저장 " 또는 그리드savecsv API 호출시 호출된다.
CALLBACK_PICKLIST_URL	픽리스트(공통코드) URL 호출전 호출되며, 조건에 따라 픽리스트 URL 을 변경해야 할 경우에 사용된다.
CALLBACK_SCREEN_URL	화면 URL 호출전 호출되며, 조건에 따라 화면 URL 을 변경해야 할 경우에 사용된다.
CALLBACK_SCRIPT_ERROR	화면내 스크립트 동작 오류 발생시 호출된다. 스크립트 실행시 try/catch 문으로 검출 가능한 오류에 대해서만 호출된다.
CALLBACK_VALIDATION_ALERT	값 검증 Alert 처리 전 호출되며, 값 검증 오류시 처리할 메시지박스 또는 내용을 제어한다.
CALLBACK_XPLUS_FILEDOWNLOAD	전용 브라우저환경에서 파일 다운로드 완료시 호출된다.
CALLBACK_XTRAN_DATALOG	XTRAN_DATALOG를 0이 아닌 값을 설정하고, XTRAN_DATALOG_SHOWTYPE 실행 파라미터를 2로 지정한 경우, XDataSet 통신 데이터 로깅 처리시 호출된다. 자세한 내용은 XTran 로깅 가이드를 참조한다.
CALLBACK_REQUESTTRAN_SENDDATAFUNC	screen requesttran API를 통해 송신 처리 전에 호출된다.
CALLBACK_REQUESTTRAN_RECVDATAFUNC	screen requesttran API를 통해 수신된 데이터 처리 전에 호출된다.
CALLBACK_SCREEN_LOADLOG	SCREEN_LOADLOG 실행 파라미터가 true이고, SCREEN_LOADLOG_SHOWTYPE 실행 파라미터를 2로 지정한 경우, 호출된다. 자세한 내용은 화면 로딩 로깅 가이드를 참조한다.

CALLBACK_BROWSER_DOCKING

```
/**
 * 실행 파라미터 - CALLBACK_BROWSER_DOCKING
 * 탭모드에서 도킹상태가 변경되었을때 호출된다. 런타임브라우저와 탭모드로 연동한 경우에 동작함
  * @param objScreen 화면 오브젝트
  * @param bDocking 도킹 여부
  */
function CallbackBrowserDocking(objScreen, bDocking) {
    console.log("CallbackScriptError> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackScriptError> bDocking = " + bDocking);
}
```

CALLBACK_EXCEL_DOWNLOAD

```
/**
 * 실행 파라미터 - CALLBACK_EXCEL_DOWNLOAD
 * 엑셀 다운로드전에 호출되는 콜백이다.
 * 엑셀 다운로드 진행시, 기본 방식인 xexcel5.jar 이용 방식을 사용하지 않고, 다른 방식으로 수행하기 위한 콜백이다.
 * (주로 XPlus5를 통한 로컬 방식의 엑셀 다운로드 수행시 사용된다.)
 * @param objScreen 화면 오브젝트
 * @param objGrid 그리드 컴포넌트 오브젝트
 * @param objExcelDownloadInfo 엑셀 다운로드 정보 오브젝트 (데이터를 포함하고 있음)
 */
function CallbackExcelDownload(objScreen, objGrid, objExcelDownloadInfo) {
    console.log("CallbackExcelDownload> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackExcelDownload> objGrid.name = " +
objGrid.getname());
    console.log(objExcelDownloadInfo);
}
```

CALLBACK_EXCEL_UPLOAD

```
/**
 * 실행 파라미터 - CALLBACK_EXCEL_UPLOAD
 * 엑셀 업로드전에 호출되는 콜백이다.
 * 엑셀 업로드 진행시, 기본 방식인 xexcel5.jar 이용 방식을 사용하지 않고 다른 방식으로 수행하기 위한 콜백이다.
 * (주로 XPlus5를 통한 로컬 방식의 엑셀 업로드 수행시 사용된다.)
 * @param objScreen 화면 오브젝트
 * @param objGrid 그리드 컴포넌트 오브젝트
 * @param objExcelUploadInfo 엑셀 업로드 정보 오브젝트 (데이터를 포함하고 있음)
 */
function CallbackExcelUpload(objScreen, objGrid, objExcelUploadInfo) {
    console.log("CallbackExcelUpload> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackExcelUpload> objGrid.name = " + objGrid.getname());
    console.log(objExcelUploadInfo);
}
```

CALLBACK_FOCUSMOVE_KEY

```
/**
 * 실행 파라미터 - CALLBACK_FOCUSMOVE_KEY
 * 키보드 입력에 의한 포커스 이동 전 처리 콜백이다.
 * 포커스 이동 관련 키 정의는 KEYMAP_FOCUS_NEXT 실행 파라미터에서 정의한다.
 * @param objScreen 화면 오브젝트
 * @param objComponent 컴포넌트 오브젝트
 * @param nInputType 컴포넌트 input_type
 * @param strValue 컴포넌트 값
 * @param isMoveNext 다음으로 이동 여부, false인 경우, 이전으로 이동
 * @param nKeyCode 포커스 이동 발생 키 코드
 * @returns
 *  true - 포커스 이동 처리 진행
 *  false - 포커스 이동 처리 중지
 */
function CallbackFocusMoveKey(objScreen, objComponent, nInputType, strValue,
isMoveNext, nKeyCode) {
    console.log("CallbackFocusMoveKey> strValue = " + strValue);

   // 아래의 샘플은 소스는 키 값이 ENTER이고, 필수 필드이고, 값의 길이가 0인 경우, 포커스 이동
중지하는 예제임
    if (nKeyCode == 13) {
     // 필수 필드이고, 값의 길이가 0인 경우, 포커스 이동 중지
        if (nInputType == 1 && strValue.length == 0) {
            return false;  // 포커스 이동 중지
        }
    }

   // 포커스 이동 진행
    return true;
}
```

CALLBACK_GRID_DOWNLOADEXCEL

```
/**
 * 실행 파라미터 - CALLBACK_GRID_DOWNLOADEXCEL
 * 그리드 "엑셀 저장" 컨텍스트 메뉴 및 downloadexcel API 콜백으로 그리드 엑셀 다운로드 옵션
파라미터를 조정하기 위해 사용한다.
 * 콜백 함수내에서 파라미터를 설정하여 downloadexcelex API를 반드시 호출해야 함
 * @param objScreen 화면 오브젝트
 * @param objGrid 그리드 컴포넌트 오브젝트
 * @param strFileName 엑셀 파일 이름
 */
function CallbackGridDownloadExcel(objScreen, objGrid, strFileName) {
    var is_save_onerow, is_include_pattern;

    console.log("CallbackGridDownloadExcel> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackGridDownloadExcel> objGrid.name = " +
objGrid.getname());
    console.log("CallbackGridDownloadExcel> strFileName = " + strFileName);
    is_save_onerow = false;
    is_include_pattern = true;

    return objGrid.downloadexcelex(strFileName, true, is_save_onerow,
        is_include_pattern, true, true,
        false, true,
        false, false);
}
```

CALLBACK_GRID_DRMFILEREAD

```
/**
 * 실행 파라미터 - CALLBACK_GRID_DRMFILEREAD
 * 전용 브라우저 환경경서 DRM_APPLY_TYPE 실행 파라미터가 2인 경우, 그리드 DRM 파일 읽기 전/
후 콜백함수
 * @param objScreen 화면 오브젝트
 * @param objGrid 그리드 컴포넌트 오브젝트
 * @param bIsBeforeRead 파일 읽기 전 여부
 * @param strFilePath 파일 경로
 * @param strCallbackFilePath bIsBeforeRead 파라미터가 true로 호출시 콜백 함수에서
리턴한 경로
 * @returns strReturnFilePath bIsBeforeRead 파라미터가 true로 호출된 경우, DRM 해
제된 파일 경로 또는 공백문자열
 */
function CallbackGridDrmFileRead(objScreen, objGrid, bIsBeforeRead,
strFilePath, strCallbackFilePath)
{
    var strReturnFilePath;

    factory.consoleprint("CallbackGridDrmFileRead> Start");
    factory.consoleprint("CallbackGridDrmFileRead> Screen URL = " +
objScreen.getscreenurl());
    factory.consoleprint("CallbackGridDrmFileRead> Grid Name = " +
objGrid.getname());
    factory.consoleprint("CallbackGridDrmFileRead> bIsBeforeRead = " +
bIsBeforeRead);
    factory.consoleprint("CallbackGridDrmFileRead> strFilePath = " +
strFilePath);
    factory.consoleprint("CallbackGridDrmFileRead> strCallbackFilePath = " +
strCallbackFilePath);

    if (bIsBeforeRead) {
        // TODO:
        // strFilePath 파라미터 값을 기준으로 DRM 해제 처리하고,
        // DRM 해제된 파일 경로 리턴, 공백 문자열 리턴시 그리드에서 strFilePath 파라미터
값을 사용함
        strReturnFilePath = "";
        return strReturnFilePath;
    }
    else {
        // TODO: strCallbackFilePath 파라미터에 해당하는 파일 삭제 처리
        return;
    }
}
```

CALLBACK_GRID_EXCEL_LOAD

```
/**
 * 실행 파라미터 - CALLBACK_GRID_EXCEL_LOAD
 * 엑셀 업로드 완료후 호출되는 콜백이다.
 * 주로 로드된 데이터를 검증하거나, 부가적인 기능을 처리하기 위한 공통 기능을 구현하기 위해 사용
된다.
 * 그리드의 getexceluploadresultinfo API를 사용하여 엑셀 업로드 상세 결과 정보를 구할 수
있다.
  * @param objScreen 화면 오브젝트
  * @param objGrid 그리드 컴포넌트 오브젝트
  */
function CallbackGridExcelLoad(objScreen, objGrid)
{
    var objResult;

    console.log("CallbackGridExcelLoad> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackGridExcelLoad> objGrid.name = " +
objGrid.getname());

    objResult = objGrid.getexceluploadresultinfo();
    console.log(objResult);

    return;
}
```

CALLBACK_GRID_PREKEYDOWN

```
/**
 * 실행 파라미터 - CALLBACK_GRID_PREKEYDOWN
 * 그리드 키 전처리 콜백이다.
 * 그리드 단축키 처리를 위한 동작을 주로 구현한다. 반드시 1 또는 0 값을 리턴해야 한다.
  * @param objScreen 화면 오브젝트
  * @param objGrid 그리드 컴포넌트 오브젝트
  * @param keycode 키 코드
  * @param bctrldown CTLR 키 눌림 상태 여부
 * @param bshiftdown SHIFT 키 눌림 상태 여부
 * @param baltdown ALT 키 눌림 상태 여부
 * @param bnumpadkey 숫자패드 키 여부
 * @param beditmode 편집모드 상태에서의 키 이벤트 발생 여부 (undefined/true/false)
 * @returns
 * 1 - 그리드에서 키 이벤트 처리 동작을 수행하지 않음
 * 0 - 그리드에서 키 이벤트 처리 동작을 수행
 */
function CallbackGridPreKeydown(objScreen, objGrid, keycode, bctrldown,
bshiftdown, baltdown, bnumpadkey, beditmode) {
    var row;

    if (beditmode === undefined) { beditmode = false; }

    console.log("CallbackGridPreKeydown> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackGridPreKeydown> objGrid.name = " +
objGrid.getname());
    console.log("CallbackGridPreKeydown> keycode = " + keycode);
    console.log("CallbackGridPreKeydown> bctrldown = " + bctrldown);
    console.log("CallbackGridPreKeydown> bshiftdown = " + bshiftdown);
    console.log("CallbackGridPreKeydown> baltdown = " + baltdown);
    console.log("CallbackGridPreKeydown> bnumpadkey = " + bnumpadkey);
    console.log("CallbackGridPreKeydown> beditmode = " + beditmode);

   // 아래는 아래의 키 처리를 위한 예시 소스임
    // Ctrl + i : 그리드 row 추가
    // Ctr + Alt + i : 선택된 row 위에 추가
    if (bctrldown && keycode == 73) {
        if (baltdown) {
            row = objGrid.getselectrow();
            row = row - 1;
            if (row < 0) { row = 0; }
            objGrid.insertitemtext(row, 0, "");
        }
        else {
            objGrid.addrow();
        }
        return 1;
    }

    return 0;
}
```

CALLBACK_GRID_SAVECSV

```
/**
 * 실행 파라미터 - CALLBACK_GRID_SAVECSV
 * 그리드 "CSV 저장" 컨텍스트 메뉴 및 savecsv API 콜백으로 그리드 CSV 다운로드 옵션 파라미터를 조정하기 위해 사용한다.
 * 콜백 함수내에서 파라미터를 설정하여 savecsvex API를 반드시 호출해야 함
 * @param objScreen 화면 오브젝트
 * @param objGrid 그리드 컴포넌트 오브젝트
 */
function CallbackGridSaveCsv(objScreen, objGrid, strFileName) {
    var csv_file_name, is_include_pattern, is_include_header,
is_close_savewnd,
        is_open_excel, is_prompt_overwrite, is_show_savewnd,
str_data_delimiter,
        is_check_data, is_include_statdata, is_save_onerow;

    console.log("CallbackGridSaveCsv> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackGridSaveCsv> objGrid.name = " + objGrid.getname());

    csv_file_name = "";
    is_include_pattern = false;
    is_include_header = true;
    is_close_savewnd = true;
    is_open_excel = false;
    is_prompt_overwrite = true;
    is_show_savewnd = true;
    str_data_delimiter = ",";
    is_check_data = true;
    is_include_statdata = true;
    is_save_onerow = true;

    return objGrid.savecsvex(csv_file_name, is_include_pattern,
is_include_header, is_close_savewnd,
        is_open_excel, is_prompt_overwrite, is_show_savewnd,
str_data_delimiter,
        is_check_data, is_include_statdata, is_save_onerow);
}
```

CALLBACK_GRID_UPLOADEXCEL

```
/**
 * 실행 파라미터 - CALLBACK_GRID_UPLOADEXCEL
 * 그리드 "엑셀 불러오기" 컨텍스트 메뉴 및 uploadexcel API 콜백으로 그리드 엑셀 업로드 옵션 파라미터를 조정하기 위해 사용한다.
 * 콜백 함수내에서 파라미터를 설정하여 uploadexcelex API를 반드시 호출해야 함
 * @param objScreen 화면 오브젝트
 * @param objGrid 그리드 컴포넌트 오브젝트
 * @param bAppendMode [옵션] 데이터 Append 모드 (기본값: false)
 * @param objFile [옵션] 업로드할 엑셀 파일에 대한 File 오브젝트 (기본값: null)
 * @param strAccept [옵션] 선택 파일 유형 문자열 (기본값: 공백문자열)
 */
function CallbackGridUploadExcel(objScreen, objGrid, bAppendMode, objFile,
strAccept) {
    var is_save_onerow, is_include_pattern;

    console.log("CallbackGridUploadExcel> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackGridUploadExcel> objGrid.name = " +
objGrid.getname());
    console.log("CallbackGridUploadExcel> bAppendMode = " + bAppendMode);
    console.log("CallbackGridUploadExcel> strAccept = " + strAccept);

    return objGrid.uploadexcelex(1, 1, 1, "A", 1, bAppendMode, "", objFile,
strAccept, true);
}
```

CALLBACK_META_DATA

```
/**
 * 실행 파라미터 - CALLBACK_META_DATA
 * 컴포넌트 생성시 메타 처리 콜백
 * 자세한 내용은 doc 폴더내 다국어 개발을 위한 가이드 문서를 참조한다.
 * @param strMetaId 메타 ID
 * @param objMetaValue 메타 ID에 해당하는 값
 * @param strScreenUrl 화면 URL
 * @param strScreenId 화면 ID
 * @param nObjectKind 오브젝트 유형 (XFD_OBJKIND_CONTROL/XFD_OBJKIND_SHAPE)
 * @param nComponentKind 오브젝트 유형 (getcontrolkind, getshapekind)
 *              또는
XFD_CTRLKIND_TABITEM/XFD_CTRLKIND_GRIDCOLUMNHEAD/XFD_CTRLKIND_GRIDCOLUMNDATA
 * @param objPropInfo 컴포넌트 속성 정보 오브젝트 (key: 속성명, value: 속성값)
 * @param strUserLanguage 사용자 실행 환경 언어
 * @param strUserCountry 사용자 실행 환경 국가
 * @returns
 *  1 - 메타 정보 처리함
 *  0 - 메타 정보 처리 하지 않음(엔진 기본 동작 수행)
 */
function CallbackMetaData(strMetaId, objMetaValue, strScreenUrl,
strScreenId, nObjectKind, nComponentKind, objPropInfo, strUserLanguage,
strUserCountry) {
    if (nObjectKind == XFD_OBJKIND_CONTROL) {
        switch (nComponentKind) {
            case XFD_CTRLKIND_HYPERTEXT:
            case XFD_CTRLKIND_PUSHBUTTON:
            case XFD_CTRLKIND_CHECKBOX:
            case XFD_CTRLKIND_RADIOBUTTON:
                if (objMetaValue["text_or_title"]) { objPropInfo["text"] =
objMetaValue["text_or_title"]; }
```

```
                if (objMetaValue["tooltip"]) { objPropInfo["tooltip"] =
objMetaValue["tooltip"]; }
                if (objMetaValue["hint_text"]) { objPropInfo["hint_text"] =
objMetaValue["hint_text"]; }
                if (objMetaValue["description"]) {
objPropInfo["description"] = objMetaValue["description"]; }
                break;

            case XFD_CTRLKIND_FIELD:
            case XFD_CTRLKIND_COMBOBOX:
            case XFD_CTRLKIND_DATE:
                if (objMetaValue["tooltip"]) { objPropInfo["text"] =
objMetaValue["tooltip"]; }
                if (objMetaValue["hint_text"]) { objPropInfo["hint_text"] =
objMetaValue["hint_text"]; }
                if (objMetaValue["description"]) {
objPropInfo["description"] = objMetaValue["description"]; }
                break;

            case XFD_CTRLKIND_MULTILINE:
                if (objMetaValue["tooltip"]) { objPropInfo["tooltip"] =
objMetaValue["tooltip"]; }
                if (objMetaValue["description"]) {
objPropInfo["description"] = objMetaValue["description"]; }
                break;

            case XFD_CTRLKIND_GRID:
                if (objMetaValue["description"]) {
objPropInfo["description"] = objMetaValue["description"]; }
                break;

            case XFD_CTRLKIND_TABITEM:
                if (objMetaValue["text_or_title"]) { objPropInfo["title"] =
objMetaValue["text_or_title"]; }
                if (objMetaValue["tooltip"]) { objPropInfo["tooltip"] =
objMetaValue["tooltip"]; }
                if (objMetaValue["description"]) {
objPropInfo["description"] = objMetaValue["description"]; }
                break;

            case XFD_CTRLKIND_GRIDCOLUMNHEAD:
                if (objMetaValue["text_or_title"]) { objPropInfo["title"] =
objMetaValue["text_or_title"]; }
                if (objMetaValue["tooltip"]) { objPropInfo["tooltip"] =
objMetaValue["tooltip"]; }
                break;

            case XFD_CTRLKIND_GRIDCOLUMNDATA:
                if (objMetaValue["hint_text"]) { objPropInfo["hint_text"] =
objMetaValue["hint_text"]; }
                if (objMetaValue["description"]) {
```

```
objPropInfo["description"] = objMetaValue["description"]; }
                break;

            default:
                break;
        }
    }
    else if (nObjectKind == XFD_OBJKIND_SHAPE) {
        switch (nComponentKind) {
            case XFD_SHAPEKIND_CAPTION:
                if (objMetaValue["text_or_title"]) { prop_info["text"] =
objMetaValue["text_or_title"]; }
                break;

            default:
                break;
        }
    }

    return 1;
}
```

CALLBACK_META_LOAD

```
/**
 * 실행 파라미터 - CALLBACK_META_LOAD
 * 메타 데이터 로드시 호출되는 콜백
 * @param strMetaUrl 메타 URL
 * @param strMetaId 메타 ID
 * @param strMetaValue 메타 ID에 해당하는 값
 * @param strUserLanguage 사용자 실행 환경 언어
 * @param strUserCountry 사용자 실행 환경 국가
 * @returns 메타 정보 오브젝트
 */
function CallbackMetaLoad(strMetaUrl, strMetaId, strMetaValue,
strUserLanguage, strUserCountry) {
    var meta_value_item_arr, meta_value_obj;

    // text/title, tooltip, hint_text, description으로 되어 있는 경우
    if (strMetaValue.indexOf("^$;") != -1) { meta_value_item_arr =
strMetaValue.split("^$;"); }
    else { meta_value_item_arr = strMetaValue.split(";"); }

    meta_value_obj = {};

    // undefined 및 공백문자열이 아닌 경우 true
    if (meta_value_item_arr[0]) { meta_value_obj["text_or_title"] =
meta_value_item_arr[0]; }
```

```
    if (meta_value_item_arr[1]) { meta_value_obj["tooltip"] =
meta_value_item_arr[1]; }
    if (meta_value_item_arr[2]) { meta_value_obj["hint_text"] =
meta_value_item_arr[2]; }
    if (meta_value_item_arr[3]) { meta_value_obj["description"] =
meta_value_item_arr[3]; }

    return meta_value_obj;
}
```

CALLBACK_PICKLIST_URL

```
/**
 * 실행 파라미터 - CALLBACK_PICKLIST_URL
 * 코드 파일을 로드하기 전에 코드 파일 URL에 대한 전 처리 콜백이다.
 * 조건에 따라 코드 파일 URL을 변경해야 할 경우에 사용된다.
 * @param strPicklistUrl 로드할 픽리스트 URL
 * @returns
 *   strReturnPicklistUrl - 로드할 픽리스트 URL (1글자 이상의 문자열)
 *   undefined | '' - 화면 URL 변경없이 사용
 */
function CallbackPicklistUrl(strPicklistUrl)
{
    console.log("strPicklistUrl = " + strPicklistUrl);
    return;
}
```

CALLBACK_SCREEN_URL

```
/**
 * 실행 파라미터 - CALLBACK_SCREEN_URL
 * 화면 파일을 로드하기 전에 화면 파일 URL에 대한 전 처리 콜백이다.
 * 조건에 따라 화면 파일 URL을 변경해야 할 경우에 사용된다.
 * @param strScreenUrl 로드할 화면 URL
 * @param nLoadType 화면 URL을 로드 하는 유형 (0: tab link screen/portlet,
1:load popup, 2:load menu, 3:load portlet)
 * @param objParentScreen 로드할 화면의 부모 화면 인스턴스 (null 또는 undefined 값
일 수 있음)
 * @param strParentScreenUrl 로드할 화면의 부모 화면 URL ("" 또는 부모 화면 URL)
 * @param objParentComponent 로드할 화면이 표시될 컴포넌트 (null 또는 부모 컴포넌트 오
브젝트)
 * @returns
 *   strReturnScreenUrl 로드할 화면 URL (1글자 이상의 문자열)
 *   undefined | '' 화면 URL 변경없이 사용
 */
```

```
function CallbackScreenUrl(strScreenUrl, nLoadType, strParentScreenUrl,
objParentScreen, objParentComponent)
{
    console.log("CallbackScreenUrl> strScreenUrl = " + strScreenUrl);
    console.log("CallbackScreenUrl> nLoadType = " + nLoadType);
    console.log("CallbackScreenUrl> strParentScreenUrl = " +
strParentScreenUrl);

    if (objParentScreen !== undefined && objParentScreen !== null) {
        console.log("CallbackScreenUrl> strParentScreenId = " +
objParentScreen.getscreenid());
    }

    if (objParentComponent !== undefined && objParentComponent !== null) {
        console.log("CallbackScreenUrl> strParentComponentName = " +
objParentComponent.getname());
    }

    return;
}
```

CALLBACK_SCRIPT_ERROR

```
/**
 * 실행 파라미터 - CALLBACK_SCRIPT_ERROR
 * 스크립트 오류 처리 콜백
 * 스크립트 실행시 try/catch 문으로 검출 가능한 오류에 대해서만 호출됨
 * @param objScreen 화면 오브젝트
 * @param strEventName 이벤트 이름
 * @param strFuncName 함수 이름
 * @param strErrorName 에러 이름
 * @param strErrMsg 에러 메시지
 * @returns
 *  1 - 엔진에서 스크립트 오류 메시지 박스를 표시하지 않음
 *    그외 - 엔진에서 오류 메시지 박스 표시
 */
function CallbackScriptError(objScreen, strEventName, strFuncName,
strErrorName, strErrMsg) {
    console.log("CallbackScriptError> objScreen.url = " +
objScreen.getscreenurl());
    console.log("CallbackScriptError> strEventName = " + strEventName);
    console.log("CallbackScriptError> strFuncName = " + strFuncName);
    console.log("CallbackScriptError> strErrorName = " + strErrorName);
    console.log("CallbackScriptError> strErrMsg = " + strErrMsg);

    // 1 - 엔진에서 스크립트 오류 메시지 박스를 표시하지 않음, 그외 - 엔진에서 오류 메시지 박스 표
시
```

```
    // return 1;
}
```

CALLBACK_VALIDATION_ALERT

```
/**
 * 실행 파라미터 - CALLBACK_VALIDATION_ALERT
 * 값 검증 Alert 처리 전 호출되며, 값 검증 오류시 처리할 메시지박스 또는 내용을 제어한다.
 * @param objScreen 화면 오브젝트
 * @param objComponent 컴포넌트 오브젝트
 * @param strComponentValue 컴포넌트 값
 * @param strValidType 검증 유형 이름
 *          'required': 필수 항목에 값이 없는 경우
 *          'length_min': 최소 길이보다 짧은 경우
 *          'length_max': 최소 길이보다 긴 경우
 *          'value_min': 최소 값보다 작은 경우
 *          'value_max': 최대 값보다 큰 경우
 *          'invalid_number': 값이 숫자가 아닌 경우
 *          'invalid_code': 유효한 코드 값이 아닌 경우
 *          'invalid_date': 유효하지 않는 날짜
 *          'invalid_keyin': 유효하지 않는 글자 입력
 *          'move_before': 첫번재 페이지 이전으로 이동시
 *          'move_after': 마지막 페이지 이후로 이동시
 *          'date_before': 유효기간 이전 날짜 선택시
 *          'date_after': 유효기간 이후 날짜 선택시
 *          'date_saturday': 토요일을 선택한 경우
 *          'date_sunday': 일요일을 선택한 경우
 *          'date_holiday': 공휴일을 선택한 경우
 *          'date_unknown': 알 수 없는 날짜 관련 오류
 * @param nEventType 검증 이벤트 유형 (2: API, 5: 포커스아웃(BLUR))
 * @returns
 *  undefined 기본 값 검증 메시지 박스 표시 (예시: 단순 return; 호출시)
 *  objRetInfo 메시지 박스 제어용 오브젝트로 아래와 같은 속성을 가짐
 *     {
 *      show_alert: alert 표시 여부 (true(기본값)/false)
 *         alert_title: alert 박스 타이틀 (변경하지 않을 경우, 공백문자열 지정)
 *         alert_message: alert 메시지 (변경하기 않을 경우, 공백 문자열 지정)
 *     }
 */
function CallbackValidationAlert(objScreen, objComponent, strComponentValue,
strValidType) {
    var i, nObjectKind, objRetetInfo;

    objRetInfo = {
        show_alert: true,
        alert_title: "",
        alert_message: ""
    };
    console.log("CallbackValidationAlert> Screen URL = [" +
objScreen.getscreenurl() + "]");
    console.log("CallbackValidationAlert> Component Name = [" +
objComponent.getname() + "]");
    console.log("CallbackValidationAlert> strComponentValue = [" +
strComponentValue + "]");
    console.log("CallbackValidationAlert> strValidType = [" + strValidType +
"]");
    console.log("CallbackValidationAlert> nEventType = [" + nEventType +
"]");

   // 아래 예시 소스는 유효하지 않은 날짜에 대한 값 검증인 경우, 값을 지우는 예시 소스입니다.
    if (strValidType == "invalid_date") {
        objComponent.settext("");
    }

    return objRetInfo;
}
```

CALLBACK_XPLUS_FILEDOWNLOAD

```
/**
 * 실행 파라미터 - CALLBACK_XPLUS_FILEDOWNLOAD
 * 전용 브라우저환경에서 파일 다운로드 완료 콜백함수
  * @param strFilePath 파일 경로
  */
function CallbackXPlusFileDownload(strFilePath)
{
    factory.consoleprint("CallbackXPlusFileDownload> Start");
    factory.consoleprint("CallbackXPlusFileDownload> strFilePath = " +
strFilePath);
}
```

CALLBACK_REQUESTTRAN_SENDDATAFUNC

```
/**
 * 실행 파라미터 - CALLBACK_REQUESTTRAN_SENDDATAFUNC
 * 화면 requesttran 송신 데이터 처리 콜백
 * 화면 requesttran API 함수 호출시 AJAX로 데이터를 송신하기 전에 호출됨
 * 화면내에서 settranmaptraninfo 함수를 통해서 송신 데이터 및 부가 정보를 설정해야 함
  * @param objScreen 화면 오브젝트
  * @param strTranMapId 트랜맵 ID
  */
function CallbackTranSendDataFunc(objScreen, strTranMapId) {
    var send_data_str, optionInfo;
    console.log(this);
    console.log("CallbackTranSendDataFunc> Screen URL = " +
objScreen.getscreenurl());
    console.log("CallbackTranSendDataFunc> strTranMapId = " + strTranMapId);

    // TODO: strTranMapId에 대한 송신 데이터 문자열 생성
    send_data_str = "senddata_" + strTranMapId;

    optionInfo = {
        tran_url: "terminal/jsp/" + strTranMapId + ".jsp",
        is_async: true,
        is_encode_url: true,
        http_header_str: "",
        timeout: 0,
        http_method: "POST"
    };

    // requesttran 송신 데이터 및 옵션 설정
    objScreen.settranmaptraninfo(strTranMapId, send_data_str, optionInfo);
}
```

CALLBACK_REQUESTTRAN_RECVDATAFUNC

```
/**
 * 실행 파라미터 - CALLBACK_REQUESTTRAN_RECVDATAFUNC
 * 화면 requesttran 수신 데이터 처리 콜백
 * 화면 on_trancomplete 이벤트가 호출된기 전에 호출됨
  * @param objScreen 화면 오브젝트
  * @param strTranMapId 트랜맵 ID
  * @param result 처리 결과 (1: 성공, 그외 오류)
  * @param recv_userheader 서버에서 수신된 사용자 정의 데이터 (기본값: "")
  * @param recv_code 서버에서 수신된 처리 결과 코드 (기본값: "")
  * @param recv_msg 서버에서 수신된 처리 결과 메시지 (기본값: "")
  * @param recv_data 서버에서 수신된 데이터
  */
function CallbackTranRecvDataFunc(objScreen, strTranMapId, result,
recv_userheader, recv_code, recv_msg, recv_data) {
    var tran_url, is_async, is_encode_url, http_header_str, send_data_str,
timeout, http_method;

    console.log(this);
    console.log("CallbackTranRecvDataFunc> Screen URL = " +
objScreen.getscreenurl());
    console.log("CallbackTranRecvDataFunc> strTranMapId = " + strTranMapId);
    console.log("CallbackTranRecvDataFunc> result = " + result);
    console.log("CallbackTranRecvDataFunc> recv_userheader = " +
recv_userheader);
    console.log("CallbackTranRecvDataFunc> recv_code = " + recv_code);
    console.log("CallbackTranRecvDataFunc> recv_msg = " + recv_msg);
    console.log("CallbackTranRecvDataFunc> recv_data = " + recv_data);

    // TODO: recv_data 파라미터 값을 기준으로 수신 데이터를 파싱하여 화면에 표시 처리
}
```

이벤트 글로벌 콜백

이벤트 글로벌 콜백은 이벤트 이름을 지정하여 해당 이벤트 발생시 지정된 콜백 이전에 호출되는 콜백이다.

구분	내용
CALLBACK_EVENT_NAME	• 콜백 호출 대상 이벤트 이름 배열을 지정한다. • 예시: ['on_fileloadstart', 'on_fileload']
CALLBACK_EVENT_BEFORE	CALLBACK_EVENT_NAME 에 지정된 이벤트 발생시 호출될 함수 이름

이벤트 글로벌 콜백 예시

```
/**
 * 실행 파라미터 - CALLBACK_EVENT_BEFORE, CALLBACK_EVENT_NAME
 * CALLBACK_EVENT_NAME 실행 파라미터에 지정된 이벤트 발생시, 화면에서 지정한 이벤트 콜백이 호출되기 전에 호출된다.
 * 콜백 함수 호출시 전달되는 파라미터는 화면, 컴포넌트, 이벤트 이름, 원래 이벤트 콜백 파라미터들(가변적임)로 구성된다.
 * 이벤트에 따라서 콜백 함수 개발시 파라미터로 전달되는 파라미터가 가변적이기 때문에,
 * 함수 내에서 arguments 오브젝트를 통해서 가변 파라미터를 처리해야 한다.
 * arguments 오브젝트는 자바스크립트 언어에서 제공하는 기본 오브젝트로,
 * 함수에 전달된 파라미터에 접근하기 위한 기능을 수행하는 특별한 배열과 비슷한 기능을 제공하는 오브젝트이다.
 * @param objScreen 화면 오브젝트
 * @param objComponent 컴포넌트 오브젝트
 * @param strEventName 이벤트 이름
 * @param ... 이벤트 발생시 전달되는 파라미터들
 * @returns
 *     함수의 리턴값이 있는 경우, 화면에 지정한 이벤트 콜백이 호출되지 않고, 리턴값이 엔진에 전달된다.
 */
function CallbackEventBefore(objScreen, objComponent, strEventName) {
    var i, argument_count;

    console.log("CallbackEventBefore> Screen URL = [" + objScreen.getscreenurl() + "]");

   // 화면인 경우, 화면 ID
    console.log("CallbackEventBefore> Component Name = [" + objComponent.getname() + "]");
    console.log("CallbackEventBefore> strEventName = [" + strEventName +
"]");
    console.log("CallbackEventBefore> object_kind = [" +
objComponent.getobjectkind() + "]");

   // 기본 파라미터 이외에 실제 이벤트 콜백 함수 파라미터를 arguments 오브젝트를 이용하여 콘
솔에 출력
    argument_count = arguments.length;
    for (i = 3; i < argument_count; i++) {
        console.log("CallbackEventBefore> arguments[" + i + "] = " +
arguments[i]);
    }
}
```

이벤트 글로벌 콜백 특징

이벤트 글로벌 콜백은 아래와 같은 특징을 가진다.

구분	내용
콜백 파리미터	• 콜백 함수 호출시 전달되는 파라미터는 화면, 컴포넌트, 이벤트 이름, 원래 이벤트 콜백 파라미터로 구성된다. • 따라서 콜백 함수 개발시 파라미터로 전달되는 파라미터가 가변적이기 때문에, 함수 내에서 arguments 오브젝트를 통해서 가변 파라미터를 처리해야한다. • arguments 오브젝트는 함수에 전달된 파라미터에 접근하기 위한 기능을 수행하는 특별한 오브젝트이며 • 자세한 설명은 'https://developer.mozilla.org ” 에서arguments 를 검색하여 참고한다.
콜백 함수 호출	이벤트 글로벌 콜백은 해당 이벤트에 대한 콜백 지정 여부와 상관없이 이벤트에 지정된 콜백이 호출되기 이전에 호출됨
적용 대상	• 이벤트 이름을 기준으로 처리되기 때문에, 모든 화면에 적용된다. • ***성능에 영향을 줄 수 있는 이벤트 (예: on_keydown)에 대해서는 글로벌콜백 함수내 동작 시간 부분에 주의해야 한다.***