```
module.exports = function(runtime, scope){
    var timers = Object.create(runtime.timers);
    var TimedTask = com.stardust.autojs.core.timing.TimedTask;
    var IntentTask = com.stardust.autojs.core.timing.IntentTask;
    var TimedTaskManager = com.stardust.autojs.core.timing.TimedTaskManager.Companion.getInstance();
    var bridges = require("__bridges__");

    scope.__asGlobal__(timers, ['setTimeout', 'clearTimeout', 'setInterval', 'clearInterval', 'setImmediate', 'clearImmediate']);

    scope.loop = function(){
        console.warn("loop() has been deprecated and has no effect. Remove it from your code.");
    }

    function parseConfig(c) {
        let config = new com.stardust.autojs.execution.ExecutionConfig();
        config.delay = c.delay || 0;
        config.interval = c.interval || 0;
        config.loopTimes = (c.loopTimes === undefined)? 1 : c.loopTimes;
        return config;
    }

    function parseDateTime(clazz, dateTime) {
        if(typeof(dateTime) == 'string') {
            return TimedTask.Companion.parseDateTime(clazz, dateTime);
        } else if(typeof(dateTime) == 'object' && dateTime.constructor === Date) {
            return TimedTask.Companion.parseDateTime(clazz, dateTime.getTime());
        } else if(typeof(dateTime) == 'number') {
            return TimedTask.Companion.parseDateTime(clazz, dateTime);
        } else {
            throw new Error("cannot parse date time: " + dateTime);
        }
    }

    function addTask(task) {
        TimedTaskManager.addTaskSync(task);
    }

    timers.addDailyTask = function (task) {
        let localTime = parseDateTime("LocalTime", task.time);
        let timedTask = TimedTask.Companion.dailyTask(localTime, files.path(task.path), parseConfig(task));
        addTask(timedTask);
        return timedTask;
    }

    var daysEn = ['sunday', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday'];
    var daysCn = ['一', '二', '三', '四', '五', '六', '日'];

    timers.addWeeklyTask = function (task) {
        let localTime = parseDateTime("LocalTime", task.time);
        let timeFlag = 0;
        for(let i = 0; i < task.daysOfWeek.length; i++) {
            let dayString = task.daysOfWeek[i];
            let dayIndex = daysEn.indexOf(dayString);
            if(dayIndex == -1) {
                dayIndex = daysCn.indexOf(dayString);
            }
            if(dayIndex == -1) {
                throw new Error('unknown day: ' + dayString);
            }
            timeFlag |= TimedTask.Companion.getDayOfWeekTimeFlag(dayIndex + 1);
        }
        let timedTask = TimedTask.Companion.weeklyTask(localTime, new java.lang.Long(timeFlag), files.path(task.path), parseConfig(task));
        addTask(timedTask);
        return timedTask;
    }

    timers.addDisposableTask = function (task) {
        let localDateTime = parseDateTime("LocalDateTime", task.date);
        let timedTask = TimedTask.Companion.disposableTask(localDateTime, files.path(task.path), parseConfig(task));
        addTask(timedTask);
        return timedTask;
    }

    timers.addIntentTask = function (task) {
        let intentTask = new IntentTask();
        intentTask.setScriptPath(files.path(task.path));
        task.action && intentTask.setAction(task.action);
        addTask(intentTask);
        return intentTask;
    }

    timers.getTimedTask = function(id) {
        return TimedTaskManager.getTimedTask(id);
    }

    timers.getIntentTask = function(id) {
        return TimedTaskManager.getIntentTask(id);
    }

    timers.removeIntentTask = function(id) {
        let task = timers.getIntentTask(id);
        return task && TimedTaskManager.removeTaskSync(task);
    }

    timers.removeTimedTask = function(id) {
        let task = timers.getTimedTask(id);
        return task && TimedTaskManager.removeTaskSync(task);
    }

    timers.queryTimedTasks = function (options, callback) {
        var sql = '';
        var args = [];
        function sqlAppend(str) {
            if(sql.length == 0) {
                sql += str;
            } else {
                sql += ' AND ' + str;
            }
            return true;
        }
        options.path && sqlAppend('script_path = ?') && args.push(options.path);
        return bridges.toArray(TimedTaskManager.queryTimedTasks(sql ? sql : null, args));
    }
    timers.queryIntentTasks = function (options, callback) {
        var sql = '';
        var args = [];
        function sqlAppend(str) {
            if(sql.length == 0) {
                sql += str;
            } else {
                sql += ' AND ' + str;
            }
            return true;
        }
        options.path && sqlAppend('script_path = ?') && args.push(options.path);
        options.action && sqlAppend('action = ?') && args.push(options.action);
        return bridges.toArray(TimedTaskManager.queryIntentTasks(sql ? sql : null, args));
    }

    return timers;
}
```