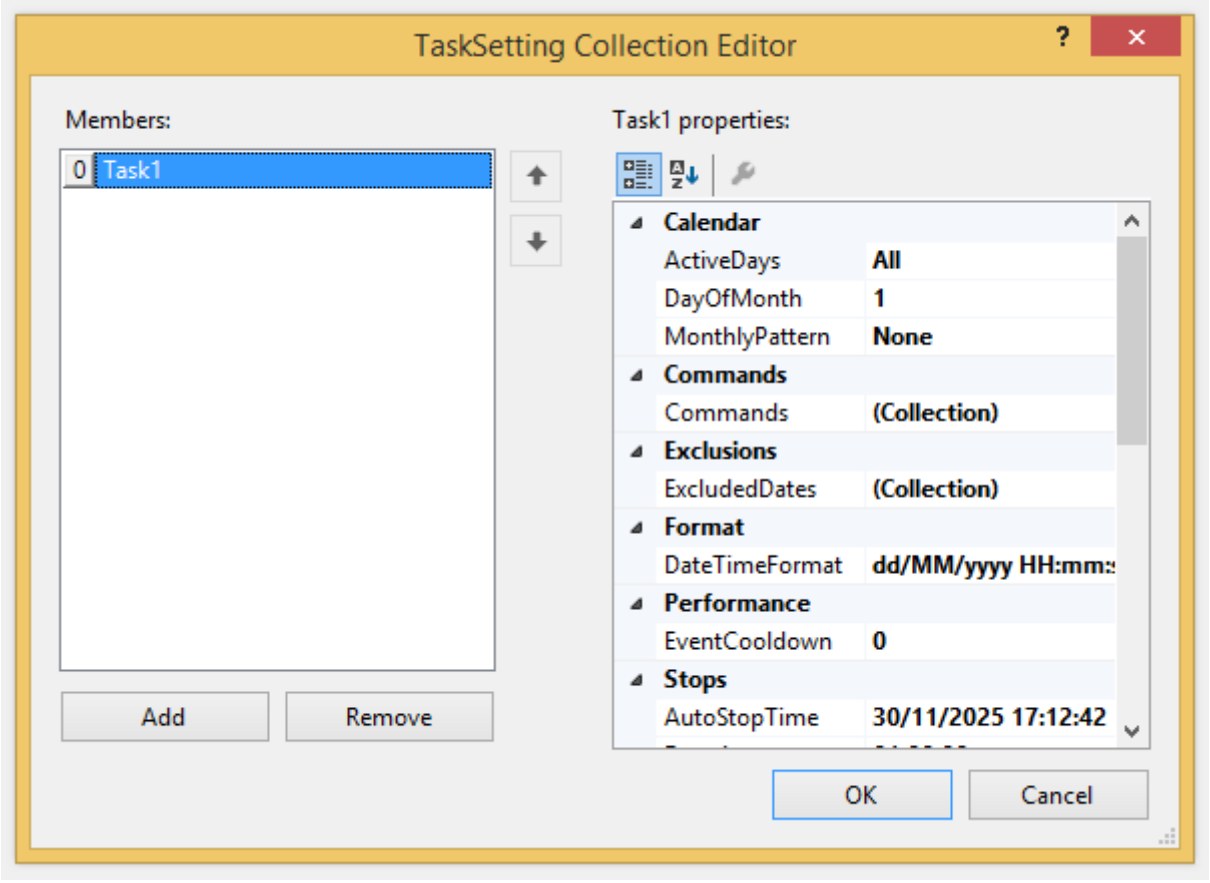
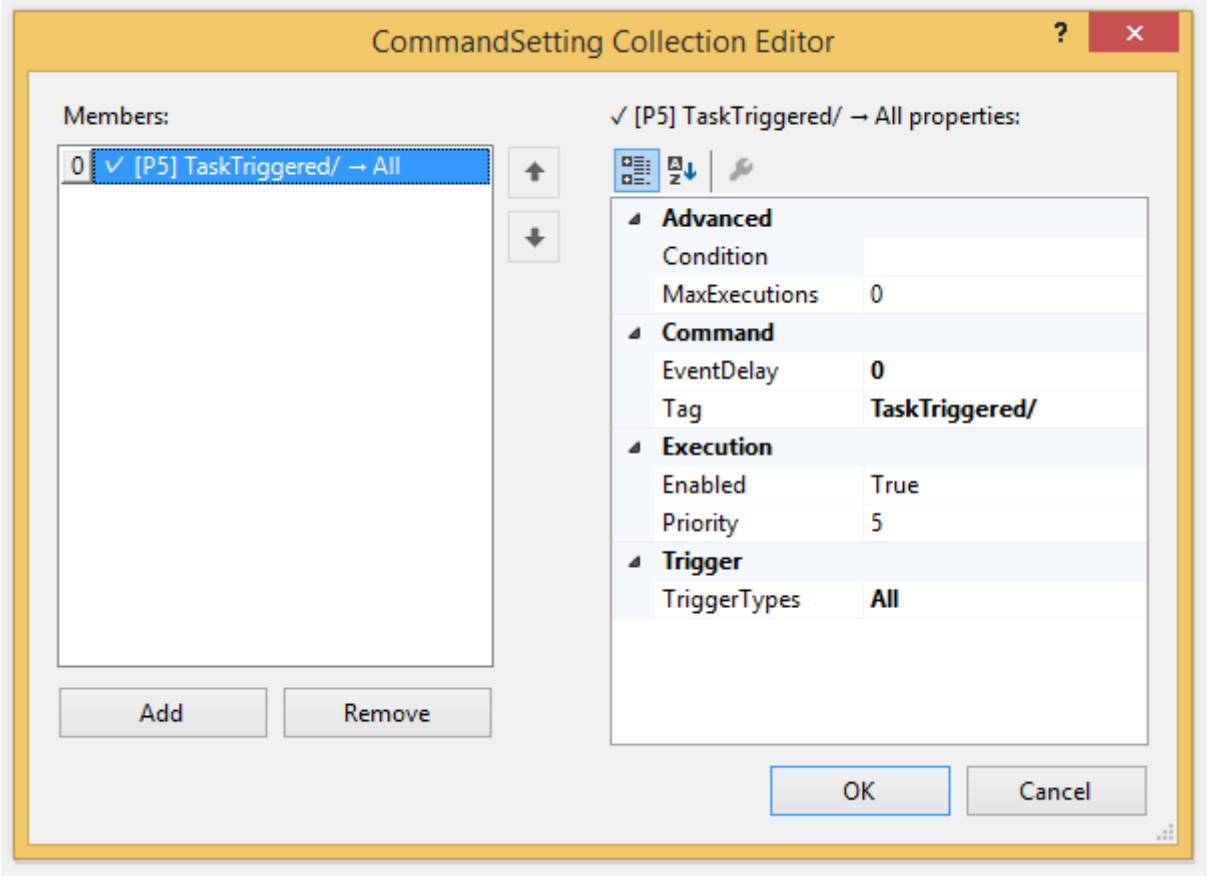
SchreinTimer





⏰ 1. Tâche simple avec déclenchement unique et simple condition

```
private void CreateSimpleTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheSimple",
        TriggerTime = DateTime.Now.AddMinutes(1), // Déclenchement dans 1 minute
        Interval = 1000, // 1 seconde entre les vérifications
        HasAutoStop = true,
        AutoStopTime = DateTime.Now.AddMinutes(2) // Arrêt après 2 minutes
    };

    // Ajouter une commande
    var command = new CommandSetting
    {
        Tag = "TaskTriggered/SchreinLED=IsOn:true;",
        EventDelay = 0,
        TriggerTypes = TriggerType.InitialTrigger,
        Priority = 1,
        //Si temp1 > temp2
        Condition = "SchreinTemperature1.Valeur > SchreinTemperature2.Valeur",
        Enabled = true

    };

    task.Commands.Add(command);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

🔄 2. Tâche récurrente avec intervalle et condition multiple

```
private void CreateRecurringTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheRécurrente",
        TriggerTime = DateTime.Now.AddSeconds(30),
        IsRecurring = true,
        IntervalUnit = TimeUnit.Minutes,
        IntervalValue = 5, // Toutes les 5 minutes
        Interval = 1000,
        ExecutionCount = 6 // S'exécute 6 fois maximum
    };

    var command = new CommandSetting
    {
        Tag = "TaskTriggered/SchreinSerial=DataToSend:[Fan|Speed|200];",
        EventDelay = 500,
        TriggerTypes = TriggerType.RecurringExecution,
        Priority = 2,
        // Pression ET température élevées
        Condition = "capteurPression1.Valeur >= 100.0 && capteurTemperature1.Valeur > 80.0",
        Enabled = true
    };

    task.Commands.Add(command);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

⌨ 3. Tâche avec dates de déclenchement multiples et Condition avec OU logique

```
private void CreateMultiDateTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheMultiDates",
        TriggerTime = DateTime.Now.AddHours(1),
        TriggerDates = new List<DateTime>
        {
            DateTime.Today.AddHours(10).AddMinutes(30), // Aujourd'hui 10:30
            DateTime.Today.AddHours(14).AddMinutes(15), // Aujourd'hui 14:15
            DateTime.Today.AddDays(1).AddHours(9),      // Demain 9:00
        },
        Interval = 1000
    };

    var command = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeMultiDates",
        EventDelay = 100,
        TriggerTypes = TriggerType.ScheduledTrigger,
        Priority = 1,
        // Capteur1 bas OU capteur2 haut
        Condition = "capteur1.Valeur < 50.0 || capteur2.Valeur > 200.0",
        Enabled = true
    };

    task.Commands.Add(command);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

🕐 4. Tâche avec fenêtres horaires

```
private void CreateTimeWindowTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheFenêtreHoraire",
        TriggerTime = DateTime.Now.AddMinutes(5),
        IsRecurring = true,
        IntervalUnit = TimeUnit.Hours,
        IntervalValue = 1, // Toutes les heures
        Interval = 1000,
        Strategy = ExecutionStrategy.TimeWindow
    };

    // Ajouter des fenêtres horaires
    task.TimeWindows.Add(new TimeWindow
    {
        StartTime = TimeSpan.FromHours(9),   // 09:00
        EndTime = TimeSpan.FromHours(12),    // 12:00
        Description = "Matin",
        ActiveDays = DayOfWeekFlags.Monday | DayOfWeekFlags.Tuesday |
                     DayOfWeekFlags.Wednesday | DayOfWeekFlags.Thursday |
                     DayOfWeekFlags.Friday,
        Enabled = true
    });

    task.TimeWindows.Add(new TimeWindow
    {
        StartTime = TimeSpan.FromHours(14), // 14:00
        EndTime = TimeSpan.FromHours(18),    // 18:00
        Description = "Après-midi",
        ActiveDays = DayOfWeekFlags.Monday | DayOfWeekFlags.Tuesday |
                     DayOfWeekFlags.Wednesday | DayOfWeekFlags.Thursday |
                     DayOfWeekFlags.Friday,
        Enabled = true
    });

    var command = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeFenêtre",
        EventDelay = 200,
        TriggerTypes = TriggerType.TimeWindowTrigger,
        Priority = 3,
        Enabled = true
    };

    task.Commands.Add(command);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

🗓️ 5. Tâche calendrier (jours spécifiques)

```
private void CreateCalendarTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheCalendrier",
        TriggerTime = DateTime.Now.AddMinutes(2),
        IsRecurring = true,
        IntervalUnit = TimeUnit.Days,
        IntervalValue = 1, // Tous les jours
        Interval = 1000,
        Strategy = ExecutionStrategy.Calendar,
        ActiveDays = DayOfWeekFlags.Monday | DayOfWeekFlags.Wednesday |
DayOfWeekFlags.Friday,
        MonthlyPattern = MonthlyPattern.FirstDay // Le premier du mois
    };

    var command = new CommandSetting
    {
        Tag = "TaskTriggered/SchreinWiFi=DataToSend:[pompe|value|On]",
        EventDelay = 150,
        TriggerTypes = TriggerType.CalendarTrigger,
        Priority = 2,
        Condition = "SchreinTank.Valeur < 30.0 && schreinSwitch1.Etat == true",
        Enabled = true
    };

    task.Commands.Add(command);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

⏱ 6. Tâche avec délais entre commandes

```
private void CreateMultiCommandTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheMultiCommandes",
        TriggerTime = DateTime.Now.AddMinutes(3),
        IsRecurring = true,
        IntervalUnit = TimeUnit.Minutes,
        IntervalValue = 10, // Toutes les 10 minutes
        Interval = 1000
    };

    // Première commande - exécution immédiate
    var command1 = new CommandSetting
    {
        Tag = "TaskTriggered/PremièreCommande",
        EventDelay = 0,
        TriggerTypes = TriggerType.InitialTrigger | TriggerType.RecurringExecution,
        Priority = 1,
        Enabled = true,
        MaxExecutions = 3 // S'exécute max 3 fois
    };

    // Deuxième commande - après 2 secondes
    var command2 = new CommandSetting
    {
        Tag = "TaskTriggered/DeuxièmeCommande",
        EventDelay = 2000, // 2 secondes de délai
        TriggerTypes = TriggerType.InitialTrigger | TriggerType.RecurringExecution,
        Priority = 2,
        Enabled = true
    };

    // Troisième commande - après 1 seconde supplémentaire
    var command3 = new CommandSetting
    {
        Tag = "TaskTriggered/TroisièmeCommande",
        EventDelay = 1000, // 1 seconde après la précédente
        TriggerTypes = TriggerType.InitialTrigger | TriggerType.RecurringExecution,
        Priority = 3,
        Enabled = true
    };

    task.Commands.Add(command1);
    task.Commands.Add(command2);
    task.Commands.Add(command3);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

7. Tâche avec différents types d'arrêt

```
private void CreateStopTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheAvecArrêts",
        TriggerTime = DateTime.Now.AddMinutes(1),
        IsRecurring = true,
        IntervalUnit = TimeUnit.Minutes,
        IntervalValue = 2, // Toutes les 2 minutes
        Interval = 1000,

        // Arrêt automatique après 10 minutes
        HasAutoStop = true,
        AutoStopTime = DateTime.Now.AddMinutes(10),

        // OU arrêt après durée
        StopAfterDuration = true,
        Duration = TimeSpan.FromMinutes(5),

        // OU arrêt après nombre d'exécutions
        ExecutionCount = 5,

        // OU arrêt à une heure spécifique
        SpecificStopTime = DateTime.Today.AddHours(17) // 17:00 aujourd'hui
    };

    // Commande normale
    var normalCommand = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeNormale",
        EventDelay = 0,
        TriggerTypes = TriggerType.All & ~TriggerType.StopTrigger, // Tous sauf StopTrigger
        Priority = 1,
        Enabled = true
    };

    // Commande d'arrêt (exécutée avant l'arrêt)
    var stopCommand = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeArrêt",
        EventDelay = 500,
        TriggerTypes = TriggerType.StopTrigger, // Uniquement à l'arrêt
        Priority = 10, // Priorité basse
        Enabled = true
    };

    task.Commands.Add(normalCommand);
    task.Commands.Add(stopCommand);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```

🔧 10. Tâche avancée avec stratégie mixte

```
private void CreateAdvancedMixedTask()
{
    var task = new TaskSetting
    {
        Name = "TâcheAvancée",
        TriggerTime = DateTime.Now.AddMinutes(1),
        IsRecurring = true,
        IntervalUnit = TimeUnit.Hours,
        IntervalValue = 2, // Toutes les 2 heures
        Interval = 1000,
        Strategy = ExecutionStrategy.Mixed, // Combinaison de stratégies

        // Configuration calendrier
        ActiveDays = DayOfWeekFlags.All & ~DayOfWeekFlags.Sunday, // Tous les jours sauf
dimanche
        MonthlyPattern = MonthlyPattern.None,

        // Exclusions de dates
        ExcludedDates = new List<DateTime>
        {
            DateTime.Today.AddDays(7), // Exclusion dans 7 jours
            DateTime.Today.AddDays(14) // Exclusion dans 14 jours
        },

        // Cooldown entre événements
        EventCooldown = 5000 // 5 secondes entre les événements
    };

    // Ajouter plusieurs fenêtres horaires
    task.TimeWindows.Add(new TimeWindow
    {
        StartTime = TimeSpan.FromHours(8),
        EndTime = TimeSpan.FromHours(12),
        Description = "Fenêtre matinale",
        ActiveDays = DayOfWeekFlags.All,
        Enabled = true
    });

    task.TimeWindows.Add(new TimeWindow
    {
        StartTime = TimeSpan.FromHours(14),
        EndTime = TimeSpan.FromHours(18),
        Description = "Fenêtre après-midi",
        ActiveDays = DayOfWeekFlags.All,
        Enabled = true
    });
```

🔧 10. Tâche avancée avec stratégie mixte(suite)

```
    // Commandes avec différents déclencheurs
    var initialCommand = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeInitiale",
        EventDelay = 0,
        TriggerTypes = TriggerType.InitialTrigger,
        Priority = 1,
        Enabled = true
    };

    var recurringCommand = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeRécurrente",
        EventDelay = 1000,
        TriggerTypes = TriggerType.RecurringExecution | TriggerType.TimeWindowTrigger,
        Priority = 2,
        Enabled = true
    };

    var calendarCommand = new CommandSetting
    {
        Tag = "TaskTriggered/CommandeCalendrier",
        EventDelay = 500,
        TriggerTypes = TriggerType.CalendarTrigger,
        Priority = 3,
        Enabled = true
    };

    task.Commands.Add(initialCommand);
    task.Commands.Add(recurringCommand);
    task.Commands.Add(calendarCommand);
    schreinTimer.Tasks.Add(task);
    schreinTimer.RepaintControl();
}
```