```
using System;
using System.Collections.Generic;

namespace Microsoft.Win32.TaskScheduler
{
    internal interface IActionModel
    {
        TaskActionType ActionType { get; }
        string Id { get; set; }
    }

    internal interface IActionCollectionModel : IList<IActionModel>,
      System.Collections.IList
    {
        bool AllowConversion { get; set; }
        string Context { get; set; }
        IActionModel Add(TaskActionType actionType);
        void ConvertUnsupportedActions();
    }

    internal interface IBootTriggerModel : ITriggerModel { }

    internal interface IComHandlerActionModel : IActionModel
    {
        Guid ClassId { get; set; }
        string Data { get; set; }
    }

    internal interface ICustomTriggerModel : ITriggerModel
    {
        string Name { get; }
        IDictionary<string, string> Properties { get; }
    }

    internal interface IDailyTriggerModel : ITriggerModel
    {
        short DaysInterval { get; set; }
    }

    internal interface IEmailActionModel : IActionModel
    {
        string Server { get; set; }
        string Subject { get; set; }
        string To { get; set; }
        string Cc { get; set; }
        string Bcc { get; set; }
        string ReplyTo { get; set; }
        string From { get; set; }
        System.Net.Mail.MailPriority Priority { get; set; }
        IDictionary<string, string> HeaderFields { get; }
        string Body { get; set; }
        string[] Attachments { get; set; }
    }

    internal interface IEventTriggerModel : ITriggerModel
    {
        string Subscription { get; set; }
        IDictionary<string, string> ValueQueries { get; }
    }

    internal interface IExecActionModel : IActionModel
    {
        string Arguments { get; set; }
        string Path { get; set; }
        string WorkingDirectory { get; set; }
    }

    internal interface IIdleSettingsModel
    {
        TimeSpan? IdleDuration { get; set; }
        bool RestartOnIdle { get; set; }
        bool StopOnIdleEnd { get; set; }
        TimeSpan? WaitTimeout { get; set; }
    }

    internal interface IIdleTriggerModel : ITriggerModel { }

    internal interface ILogonTriggerModel : ITriggerModel
    {
        string UserId { get; set; }
    }

    internal interface IMaintenanceSettingsModel
    {
        TimeSpan? Deadline { get; set; }
        bool Exclusive { get; set; }
        TimeSpan? Period { get; set; }
    }

    internal interface IMonthlyDOWTriggerModel : ITriggerModel
    {
        DaysOfTheWeek DaysOfWeek { get; set; }
        MonthsOfTheYear MonthsOfYear { get; set; }
        bool RunOnLastWeekOfMonth { get; set; }
        WhichWeek WeeksOfMonth { get; set; }
    }

    internal interface IMonthlyTriggerModel : ITriggerModel
    {
        byte[] DaysOfMonth { get; set; }
        MonthsOfTheYear MonthsOfYear { get; set; }
        bool RunOnLastDayOfMonth { get; set; }
    }

    internal interface INetworkSettingsModel
    {
        Guid Id { get; set; }
        string Name { get; set; }
    }

    internal interface IPrincipalModel
    {
        string DisplayName { get; set; }
        string GroupId { get; set; }
        string Id { get; set; }
        string UserId { get; set; }
    }

    internal interface IRegisteredTaskModel
    {
        ITaskDefinitionModel Definition { get; }
        bool Enabled { get; set; }
        ITaskFolderModel Folder { get; }
        DateTime? LastRunTime { get; }
        int? LastTaskResult { get; }
        string Name { get; }
        DateTime? NextRunTime { get; }
        int? NumberOfMissedRuns { get; }
        string Path { get; }
        TaskState State { get; }
        string Xml { get; }
        IReadOnlyList<IRunningTaskModel> GetInstances();
        IReadOnlyList<DateTime> GetRunTimes(DateTime start, DateTime end, uint 
          count);
        string GetSecurityDescriptor(System.Security.AccessControl.SecurityInfos 
          includeSections);
        IRunningTaskModel Run(params string[] parameters);
        IRunningTaskModel RunEx(string[] parameters, TaskRunFlags flags, int 
          sessionId, string user);
        void SetSecurityDescriptor(string sddl, TaskSetSecurityOptions options);
        void Stop();
    }

    internal interface IRegistrationTriggerModel : ITriggerModel { }

    internal interface IRunningTaskModel
    {
        string CurrentAction { get; }
        uint? EnginePID { get; }
        Guid? InstanceGuid { get; }
        string Name { get; }
        string Path { get; }
        TaskState State { get; }
        void Refresh();
        void Stop();
    }

    internal interface ISessionStateChangeTriggerModel : ITriggerModel
    {
        TaskSessionStateChangeType StateChange { get; set; }
        string UserId { get; set; }
    }

    internal interface IShowMessageActionModel : IActionModel
    {
        string MessageBody { get; set; }
        string Title { get; set; }
    }

    internal interface ITaskDefinitionModel
    {
        IActionCollectionModel Actions { get; }
        string Author { get; set; }
        string Data { get; set; }
        DateTime? Date { get; set; }
        string Description { get; set; }
        string Documentation { get; set; }
        TaskLogonType LogonType { get; set; }
        IPrincipalModel Principal { get; }
        TaskRunLevel RunLevel { get; set; }
        string SecurityDescriptor { get; set; }
        ITaskSettingsModel Settings { get; }
        string Source { get; set; }
        ITriggerCollectionModel Triggers { get; }
        string URI { get; set; }
        Version Version { get; set; }
        string XmlText { get; set; }
    }

    internal interface ITaskFolderModel
    {
        string Name { get; }
        string Path { get; }
        ITaskFolderModel CreateFolder(string subFolderName, string sddl);
        void DeleteFolder(string subFolderName);
        void DeleteTask(string Name);
        ITaskFolderModel GetFolder(string path);
        ICollection<ITaskFolderModel> GetFolders();
        string GetSecurityDescriptor(System.Security.AccessControl.SecurityInfos ↩
          includeSections);
        IRegisteredTaskModel GetTask(string path);
        IReadOnlyList<IRegisteredTaskModel> GetTasks(bool includeHidden = false);
        IRegisteredTaskModel RegisterTask(string path, string xmlText,          ↩
          TaskCreation flags, string userId, string password, TaskLogonType     ↩
          logonType, string sddl);
        IRegisteredTaskModel RegisterTaskDefinition(string path,                ↩
          ITaskDefinitionModel pDefinition, TaskCreation flags, string userId,  ↩
          string password, TaskLogonType logonType, string sddl);
        void SetSecurityDescriptor(string sddl, TaskSetSecurityOptions options);
    }

    internal interface ITaskServiceModel
    {
        bool Connected { get; }
        string ConnectedDomain { get; }
        string ConnectedUser { get; }
        Version HighestVersion { get; }
        ITaskFolderModel RootFolder { get; }
        string TargetServer { get; }
        void Connect(string serverName, string user, string domain, string
          password);
        ITaskFolderModel GetFolder(string path);
        IReadOnlyList<IRunningTaskModel> GetRunningTasks(bool includeHidden =
          false);
        IRegisteredTaskModel GetTask(string fullPath);
        ITaskDefinitionModel NewTask();
    }

    internal interface ITaskSettingsModel
    {
        bool AllowDemandStart { get; set; }
        bool AllowHardTerminate { get; set; }
        TaskCompatibility Compatibility { get; set; }
        TimeSpan? DeleteExpiredTaskAfter { get; set; }
        bool DisallowStartIfOnBatteries { get; set; }
        bool DisallowStartOnRemoteAppSession { get; set; }
        bool Enabled { get; set; }
        TimeSpan? ExecutionTimeLimit { get; set; }
        bool Hidden { get; set; }
        IIdleSettingsModel IdleSettings { get; }
        IMaintenanceSettingsModel MaintenanceSettings { get; }
        TaskInstancesPolicy MultipleInstances { get; set; }
        INetworkSettingsModel NetworkSettings { get; }
        System.Diagnostics.ProcessPriorityClass Priority { get; set; }
        int? RestartCount { get; set; }
        TimeSpan? RestartInterval { get; set; }
        bool RunOnlyIfIdle { get; set; }
        bool RunOnlyIfNetworkAvailable { get; set; }
        bool StartWhenAvailable { get; set; }
        bool StopIfGoingOnBatteries { get; set; }
        bool UseUnifiedSchedulingEngine { get; set; }
        bool Volatile { get; set; }
        bool WakeToRun { get; set; }
    }

    internal interface ITimeTriggerModel : ITriggerModel { }

    internal interface ITriggerCollectionModel : IList<ITriggerModel>,
      System.Collections.IList
    {
        ITriggerModel AddNew(TaskTriggerType actionType);
    }

    internal interface ITriggerModel
    {
        TimeSpan? Delay { get; set; }
        bool Enabled { get; set; }
        DateTime? EndBoundary { get; set; }
        TimeSpan? ExecutionTimeLimit { get; set; }
        string Id { get; set; }
        TimeSpan? RepetitionDuration { get; set; }
        TimeSpan? RepetitionInterval { get; set; }
        bool RepetitionStopAtDurationEnd { get; set; }
        DateTime StartBoundary { get; set; }
        TaskTriggerType TriggerType { get; }
    }

    internal interface IWeeklyTriggerModel : ITriggerModel
    {
        DaysOfTheWeek DaysOfWeek { get; set; }
        short WeeksInterval { get; set; }
    }
}
```