

```
1 using System;
2 using System.Collections.Generic;
3
4 namespace Microsoft.Win32.TaskScheduler
5 {
6     internal interface IActionModel
7     {
8         TaskActionType ActionType { get; }
9         string Id { get; set; }
10    }
11
12    internal interface IActionCollectionModel : IList<IActionModel>,
13        System.Collections.IList
14    {
15        bool AllowConversion { get; set; }
16        string Context { get; set; }
17        IActionModel Add(TaskActionType actionType);
18        void ConvertUnsupportedActions();
19    }
20
21    internal interface IBootTriggerModel : ITriggerModel { }
22
23    internal interface IComHandlerActionModel : IActionModel
24    {
25        Guid ClassId { get; set; }
26        string Data { get; set; }
27    }
28
29    internal interface ICustomTriggerModel : ITriggerModel
30    {
31        string Name { get; }
32        IDictionary<string, string> Properties { get; }
33    }
34
35    internal interface IDailyTriggerModel : ITriggerModel
36    {
37        short DaysInterval { get; set; }
38    }
39
40    internal interface IEmailActionModel : IActionModel
41    {
42        string Server { get; set; }
43        string Subject { get; set; }
44        string To { get; set; }
45        string Cc { get; set; }
46        string Bcc { get; set; }
47        string ReplyTo { get; set; }
48        string From { get; set; }
49        System.Net.Mail.MailPriority Priority { get; set; }
50        IDictionary<string, string> HeaderFields { get; }
51        string Body { get; set; }
52        string[] Attachments { get; set; }
53    }
54 }
```

```
52     }
53
54     internal interface IEventTriggerModel : ITriggerModel
55     {
56         string Subscription { get; set; }
57         IDictionary<string, string> ValueQueries { get; }
58     }
59
60     internal interface IExecActionModel : IActionModel
61     {
62         string Arguments { get; set; }
63         string Path { get; set; }
64         string WorkingDirectory { get; set; }
65     }
66
67     internal interface IIIdleSettingsModel
68     {
69         TimeSpan? IdleDuration { get; set; }
70         bool RestartOnIdle { get; set; }
71         bool StopOnIdleEnd { get; set; }
72         TimeSpan? WaitTimeout { get; set; }
73     }
74
75     internal interface IIIdleTriggerModel : ITriggerModel { }
76
77     internal interface ILogonTriggerModel : ITriggerModel
78     {
79         string UserId { get; set; }
80     }
81
82     internal interface IMaintenanceSettingsModel
83     {
84         TimeSpan? Deadline { get; set; }
85         bool Exclusive { get; set; }
86         TimeSpan? Period { get; set; }
87     }
88
89     internal interface IMonthlyDOWTriggerModel : ITriggerModel
90     {
91         DaysOfTheWeek DaysOfWeek { get; set; }
92         MonthsOfTheYear MonthsOfYear { get; set; }
93         bool RunOnLastWeekOfMonth { get; set; }
94         WhichWeek WeeksOfMonth { get; set; }
95     }
96
97     internal interface IMonthlyTriggerModel : ITriggerModel
98     {
99         byte[] DaysOfMonth { get; set; }
100        MonthsOfTheYear MonthsOfYear { get; set; }
101        bool RunOnLastDayOfMonth { get; set; }
102    }
103
```

```
104     internal interface INetworkSettingsModel
105     {
106         Guid Id { get; set; }
107         string Name { get; set; }
108     }
109
110     internal interface IPrincipalModel
111     {
112         string DisplayName { get; set; }
113         string GroupId { get; set; }
114         string Id { get; set; }
115         string UserId { get; set; }
116     }
117
118     internal interface IRegisteredTaskModel
119     {
120         ITaskDefinitionModel Definition { get; }
121         bool Enabled { get; set; }
122         ITaskFolderModel Folder { get; }
123         DateTime? LastRunTime { get; }
124         int? LastTaskResult { get; }
125         string Name { get; }
126         DateTime? NextRunTime { get; }
127         int? NumberOfMissedRuns { get; }
128         string Path { get; }
129         TaskState State { get; }
130         string Xml { get; }
131         IReadOnlyList<IRunningTaskModel> GetInstances();
132         IReadOnlyList<DateTime> GetRunTimes(DateTime start, DateTime end, uint ↗
            count);
133         string GetSecurityDescriptor(System.Security.AccessControl.SecurityInfos ↗
            includeSections);
134         IRunningTaskModel Run(params string[] parameters);
135         IRunningTaskModel RunEx(string[] parameters, TaskRunFlags flags, int ↗
            sessionId, string user);
136         void SetSecurityDescriptor(string sddl, TaskSetSecurityOptions options);
137         void Stop();
138     }
139
140     internal interface IRegistrationTriggerModel : ITriggerModel { }
141
142     internal interface IRunningTaskModel
143     {
144         string CurrentAction { get; }
145         uint? EnginePID { get; }
146         Guid? InstanceGuid { get; }
147         string Name { get; }
148         string Path { get; }
149         TaskState State { get; }
150         void Refresh();
151         void Stop();
152     }
```

```
153
154     internal interface ISessionStateChangeTriggerModel : ITriggerModel
155     {
156         TaskSessionStateChangeType StateChange { get; set; }
157         string UserId { get; set; }
158     }
159
160     internal interface IShowMessageActionModel : IActionModel
161     {
162         string MessageBody { get; set; }
163         string Title { get; set; }
164     }
165
166     internal interface ITaskDefinitionModel
167     {
168         IActionCollectionModel Actions { get; }
169         string Author { get; set; }
170         string Data { get; set; }
171         DateTime? Date { get; set; }
172         string Description { get; set; }
173         string Documentation { get; set; }
174         TaskLogonType LogonType { get; set; }
175         IPrincipalModel Principal { get; }
176         TaskRunLevel RunLevel { get; set; }
177         string SecurityDescriptor { get; set; }
178         ITaskSettingsModel Settings { get; }
179         string Source { get; set; }
180         ITriggerCollectionModel Triggers { get; }
181         string URI { get; set; }
182         Version Version { get; set; }
183         string XmlText { get; set; }
184     }
185
186     internal interface ITaskFolderModel
187     {
188         string Name { get; }
189         string Path { get; }
190         ITaskFolderModel CreateFolder(string subFolderName, string sddl);
191         void DeleteFolder(string subFolderName);
192         void DeleteTask(string Name);
193         ITaskFolderModel GetFolder(string path);
194         ICollection<ITaskFolderModel> GetFolders();
195         string GetSecurityDescriptor(System.Security.AccessControl.SecurityInfos ↗
196             includeSections);
197         IRegisteredTaskModel GetTask(string path);
198         IReadOnlyList<IRegisteredTaskModel> GetTasks(bool includeHidden = false);
199         IRegisteredTaskModel RegisterTask(string path, string xmlText, ↗
200             TaskCreation flags, string userId, string password, TaskLogonType ↗
201             logonType, string sddl);
202         IRegisteredTaskModel RegisterTaskDefinition(string path, ↗
203             ITaskDefinitionModel pDefinition, TaskCreation flags, string userId, ↗
204             string password, TaskLogonType logonType, string sddl);
```

```
200     void SetSecurityDescriptor(string sddl, TaskSetSecurityOptions options);
201 }
202
203 internal interface ITaskServiceModel
204 {
205     bool Connected { get; }
206     string ConnectedDomain { get; }
207     string ConnectedUser { get; }
208     Version HighestVersion { get; }
209     ITaskFolderModel RootFolder { get; }
210     string TargetServer { get; }
211     void Connect(string serverName, string user, string domain, string password);
212     ITaskFolderModel GetFolder(string path);
213     IReadOnlyList<IRunningTaskModel> GetRunningTasks(bool includeHidden = false);
214     IRegisteredTaskModel GetTask(string fullPath);
215     ITaskDefinitionModel NewTask();
216 }
217
218 internal interface ITaskSettingsModel
219 {
220     bool AllowDemandStart { get; set; }
221     bool AllowHardTerminate { get; set; }
222     TaskCompatibility Compatibility { get; set; }
223     TimeSpan? DeleteExpiredTaskAfter { get; set; }
224     bool DisallowStartIfOnBatteries { get; set; }
225     bool DisallowStartOnRemoteAppSession { get; set; }
226     bool Enabled { get; set; }
227     TimeSpan? ExecutionTimeLimit { get; set; }
228     bool Hidden { get; set; }
229     IIdleSettingsModel IdleSettings { get; }
230     IMaintenanceSettingsModel MaintenanceSettings { get; }
231     TaskInstancesPolicy MultipleInstances { get; set; }
232     INetworkSettingsModel NetworkSettings { get; }
233     System.Diagnostics.ProcessPriorityClass Priority { get; set; }
234     int? RestartCount { get; set; }
235     TimeSpan? RestartInterval { get; set; }
236     bool RunOnlyIfIdle { get; set; }
237     bool RunOnlyIfNetworkAvailable { get; set; }
238     bool StartWhenAvailable { get; set; }
239     bool StopIfGoingOnBatteries { get; set; }
240     bool UseUnifiedSchedulingEngine { get; set; }
241     bool Volatile { get; set; }
242     bool WakeToRun { get; set; }
243 }
244
245 internal interface ITimeTriggerModel : ITriggerModel { }
246
247 internal interface ITriggerCollectionModel : IList<ITriggerModel>,
248     System.Collections.IList
```

```
249     ITriggerModel AddNew(TaskTriggerType actionType);
250 }
251
252 internal interface ITriggerModel
253 {
254     TimeSpan? Delay { get; set; }
255     bool Enabled { get; set; }
256     DateTime? EndBoundary { get; set; }
257     TimeSpan? ExecutionTimeLimit { get; set; }
258     string Id { get; set; }
259     TimeSpan? RepetitionDuration { get; set; }
260     TimeSpan? RepetitionInterval { get; set; }
261     bool RepetitionStopAtDurationEnd { get; set; }
262     DateTime StartBoundary { get; set; }
263     TaskTriggerType TriggerType { get; }
264 }
265
266 internal interface IWeeklyTriggerModel : ITriggerModel
267 {
268     DaysOfTheWeek DaysOfWeek { get; set; }
269     short WeeksInterval { get; set; }
270 }
271 }
```