Scripting API

AzureSkyController class:

SetNewWeatherProfile()

```
public void SetNewWeatherProfile(int index)
Changes the global weather with a smooth transition.
index: The target profile number in the "Global Weather Profiles" list. Set the index to -1 if you
want to reset the global weather back to the default day profile.
```

SetNewDayProfile()

```
public void SetNewDayProfile(AzureSkyProfile profile)
Changes the current day profile without transition.
```

```
public void SetNewDayProfile(AzureSkyProfile profile, float transitionTime)
Changes the current day profile and perform a transition.
```

GetOutputFloatValue()

```
public float GetOutputFloatValue(int index)
Gets the output value of Slider or Curve type properties. The “index” parameter is the element number
of the output in the list.
```

GetOutputColorValue()

```
public Color GetOutputColorValue(int index)
Gets the output value of Color or Gradient type properties. The “index” parameter is the element
number of the output in the list.
```

AzureSkyController.settings

```
public AzureSkySettings settings = new AzureSkySettings();
Stores the current sky settings after performing all profile blend transitions. Useful if you want
check any sky setting value.
```

AzureTimeController class:

UpdateCalendar()

```
public void UpdateCalendar()
Adjust the calendar when there is a change in the date. Always call this after changing the date by
script.
```

GetDayOfWeek()

```
public int GetDayOfWeek()
Gets the current day of the week and return an integer between 0 and 6.
```

```
public int GetDayOfWeek(int year, int month, int day)
Get the day of the week from a custom date and return an integer between 0 and 6.
```

GetDayOfWeekString()

```
public string GetDayOfWeekString()
Gets the current day of the week and return as string.
```

```
public string GetDayOfWeekString(int year, int month, int day)
Gets the day of the week from a custom date and return as string.
```

SetTimeline()

```
public void SetTimeline(float value)
Set the timeline using a float as parameter.
```

```
public void SetTimeline(Slider value)
Set the timeline using a slider as parameter.
```

SetNewDate()

```
public void SetNewDate(int year, int month, int day)
Sets a new custom date.
```

SetNewYear()

```
public void SetNewYear(int year)
Sets a new custom year.
```

SetNewMonth()

```
public void SetNewMonth(int month)
Sets a new custom month.
```

SetNewDay()

```
public void SetNewDay(int day)
Sets a new custom day.
```

IncreaseYear()

```
public void IncreaseYear()
Changes the date to the next year.
```

IncreaseMonth()

```
public void IncreaseMonth()
Changes the date to the next month.
```

IncreaseDay()

```
public void IncreaseDay()
Changes the date to the next day.
```

DecreaseYear()

```
public void DecreaseYear()
Changes the date to the previous year.
```

DecreaseMonth()

```
public void DecreaseMonth()
Changes the date to the previous month.
```

DecreaseDay()

```
public void DecreaseDay()
Changes the date to the previous day.
```

GetDateString()

```
public string GetDateString()
Returns the current date converted to string using the default date format used by Azure.
Format: "MMMM dd, yyyy"
```

```
public string GetDateString(string format)
Returns the current date converted to string using a custom date format.
```

Making a Timeline Transition

There are several variations of the same method to make it easier to create a timeline transition (fast progression from one time of day to another).

StartTimelineTransition ()

```
public void StartTimelineTransition(float source, float destination, float transitionTime)
Starts a timeline transition from source time of day to a destination time of day that last a period
of time in seconds.
```

```
public void StartTimelineTransition(float destination, float transitionTime)
The same as above, but the source is automatically set by taking the current time of day.
```

```
public void StartTimelineTransition(float destination)
This variation contains only one parameter (the destination time of day), this is ideal for use with
Unity's events that only allow us to use methods with one parameter, the source is automatically set
by taking the current time of day. Since the transition time is not set, you must first call the
```

```
public void StartTimelineTransition()
This variation starts a timeline transition, but does not set any parameters. You must
first call the following methods for everything to work properly.
```

```
public void SetTimelineSourceTransitionTime(float source)
```

```
public void SetTimelineDestinationTransitionTime(float destination)
```

```
public void SetTimelineTransitionTime(float transitionTime)
```

AzureEffectsController class:

InstantiateThunderEffect()

```
public void InstantiateThunderEffect(int index)
Create a thunder effect in the scene using the settings and position from the “Thunder List”. When the
thunder sound is over, the instance is automatically deleted.
```

```
public void InstantiateThunderEffect(int index, Vector3 worldPos)
Create a thunder effect in the scene using the settingsfrom the “Thunder List”. When the thunder sound
is over, the instance is automatically deleted.
```