CgiRemoteControl

CgiRemoteControl.exe is a command-line launcher that reads a SceneComposer solution (*.scs*) and a configuration XML, and then triggers and controls plugin execution. Because it lets you run plugin processing without starting or operating SceneComposer itself, it’s well-suited for automation (batch/CI) and for consistent control from external tools.

Preparation

CgiRemoteControl.exe is a **command-line tool** that loads a SceneComposer solution and performs processing according to a configuration XML. The concrete processing is implemented on the **plugin** side. When necessary, pass a string command with `--PlugInControl` to invoke the plugin's `OnPluginControl(string)`. Results are shown in the console output and by the process **exit code**.

Prerequisites

- .NET Framework **4.8.0 or later**
- Required files: `SCHost.dll`, `CgiRemoteControlConfig.xml`

File Placement

- Place CgiRemoteControl.exe in the **same directory** as SceneComposer.exe.
- Place the configuration file CgiRemoteControlConfig.xml in the same directory, or **specify it as an argument** when running the tool.
- Paths may be **relative or absolute**.

Example Layout

```
bin/
 ├─ SceneComposer.exe
 ├─ SCHost.dll
 ├─ CgiRemoteControl.exe
 └─ CgiRemoteControlConfig.xml
```

Configuration File

CgiRemoteControl.exe reads settings from an **XML** configuration file. Use the file name **CgiRemoteControlConfig.xml**. By specifying the location of `SCHost.dll` and the allowed wait time, the host is resolved at runtime and a timeout threshold for long-running operations is defined. If you use the default name, follow the placement above; if you use a different name or path, pass it as an argument when running the tool.

Minimal Sample

- **`SCHostPath`**: Path to `SCHost.dll`, relative to the tool's directory
- **`TimeoutMilliseconds`**: Timeout in milliseconds

```
<CgiRemoteControlConfig>
  <SCHostPath>C:/path/SCHost.dll</SCHostPath>
  <TimeoutMilliseconds>60000</TimeoutMilliseconds>
</CgiRemoteControlConfig>
```

Command-line usage

Below are the basic steps for running **CgiRemoteControl.exe**. Because **CgiRemoteControl.exe** only triggers plugin processing, a separate **plugin** implementation is required.

Execution Format

Run **CgiRemoteControl.exe** from a script or command prompt using the following format. See the next section for the runtime **arguments**.

```
CgiRemoteControl.exe --SolutionFilePath "<path>.scs" [--PlugInControl "<command>"]
[<configFile>]
```

Arguments

Specifying the Solution Path (Required)

`--SolutionFilePath <path>`
Specifies the path to the `.scs` solution to load (relative or absolute path is acceptable).
Example:

```
--SolutionFilePath "C:/path/test.scs"
```

Specifying the Plugin Command (Optional)

`--PlugInControl "<command>"`
Specifies the **plugin command** to execute during processing. If omitted, the plugin's `OnPluginControl` implementation is **not invoked**.
Example:

```
--PlugInControl "BuildHost 4711"
```

Specifying the Configuration File (Optional)

`<configFile>`
Specifies the XML **configuration file name** to use. If omitted, the default **CgiRemoteControlConfig.xml** is used.
Example:

```
C:/path/CgiRemoteControlConfig.xml
```

Checking Execution Logs

The console prints the selected configuration file name, resolved `SCHostPath`, `SolutionFilePath`, `PlugInControl`, `TimeoutMilliseconds`, and so on. When processing completes, a final message is printed. You can check the process exit code with `echo %ERRORLEVEL%`.

Exit Code

Return codes indicate success or failure (0 for success; non-zero for failure).

Code	Description
0	Success
-1	Configuration file not found
-2	Permission error accessing the configuration file
-3	Invalid configuration file format
-4	Error loading the solution
-5	SCHost not found
-6	Process timed out
-7	Process cancelled
-8	General error

Code Execution Example

```
D:/work/bin/SceneComposer>CgiRemoteControl.exe --SolutionFilePath "C:/path/test.scs" --
PlugInControl "BuildHost 4711"
[2025-01-17 10:42:05.954] [Info] Using configuration file: CgiRemoteControlConfig.xml
[2025-01-17 10:42:06.033] [Info] Resolved SCHostPath: C:/path/SCHost.dll
[2025-01-17 10:42:06.033] [Info] Loaded configuration:
[2025-01-17 10:42:06.033] [Info]   SCHostPath: C:/path/SCHost.dll
[2025-01-17 10:42:06.033] [Info]   SolutionFilePath: C:/path/test.scs
```

```
[2025-01-17 10:42:06.033] [Info]   PlugInControl: BuildHost 4711
[2025-01-17 10:42:06.033] [Info]   TimeoutMilliseconds: 1800000
[2025-01-17 10:42:06.033] [Info] CgiRemoteControl Process start.
Info: The following licenses were acquired successfully: 2D, 3D, Globalization, SmartImporter,
AssetLibraryVerification, Safety.
0000000.574 [0x00009BC0] WARN  Controls      Failed to create source stream object. Possibly
no VideoStreamer is registered. {VideoStreamBehavior.cpp(143): VideoStreamBehavior}
Info: Starting schema validation of file:
Info: 1/17/2025 10:42:15 AM:OnSolutionChanged
Warning: The application specification file is missing or invalid!
[2025-01-17 10:42:15.021] [Info] Solution loaded successfully.
Info: 1/17/2025 10:42:15 AM:test log start
Info: 1/17/2025 10:42:15 AM:OnPluginControl BuildHost 4711 Processing...10%
Info: 1/17/2025 10:42:16 AM:OnPluginControl BuildHost 4711 Processing...50%
Info: 1/17/2025 10:42:17 AM:OnPluginControl BuildHost 4711 Processing...100%
Info: 1/17/2025 10:42:17 AM:test log end
[2025-01-17 10:42:17.063] [Info] Plugin executed successfully.
[2025-01-17 10:42:17.063] [Info] CgiRemoteControl Process end.

>echo %ERRORLEVEL%
0
```

Plugin-side implementation

CgiRemoteControl.exe is the execution trigger that starts plugin processing from the command line; the actual processing is implemented and executed on the plugin side. Therefore, on the plugin side, implement the **IEventListener** interface to handle events.

- Set the **EventsToSubscribe** property to include **EventId.PluginControlCommand**. If this is not set, **OnPluginControl** will not be called.
- The **OnPluginControl** method processes the received command string and returns a status code indicating success or failure.
- If **OnPluginControl** returns a non-zero value, subsequent calls to **OnPluginControl** in other plugins will not occur.

This design ensures that when the `--PlugInControl` argument is specified, the plugin is invoked correctly.

> See the Sample Implementation Guide for step-by-step instructions and the complete source code.

Implementation Example:

```
public class SamplePlugin : IEventListener
{
    // Subscribe to specific events by implementing this property
    public virtual EventId EventsToSubscribe
    {
        get { return EventId.PluginControlCommand; }
    }

    // Implement the OnPluginControl method to handle the plugin command
    public int OnPluginControl(string command)
    {
        //this.LoggerSvc.Info(DateTime.Now + ":test log start"); // Info: 1/17/2025 11:13:01
AM:test log start
        //Console.WriteLine($"Received command: {command}");     // Received command:
BuildHost 4711
        // Perform actions based on the command
        return 0; // Return an appropriate status code
    }
```

```
}
```

Sample Implementation Guide

This page explains the implementation steps and sample code for the SampleCgiRemoteControl plugin as a concrete example of running a plugin from CgiRemoteControl.

> For an overview of CgiRemoteControl.exe and its command-line options, see Command-line usage. For a summary of the plugin-side interface specification, see Plugin-side Implementation.

Plugin class

First, create a class that implements IEventListener.

```
public class SampleCgiRemoteControl : IEventListener {


}
```

Implementing IPlugin

Next, implement the members of the IPluginService interface.

```
public string Description
{
    get { return "A plugin running on the CgiRemoteControls"; }
}
public IEnumerable<int> FeatureIds { get { return null; } }
public string Name
{
    get { return "Sample CgiRemoteControl plugin"; }
}
public string Version
{
    get { return "1.0.0"; }
}
```

Implementing IEventListener event handlers

Then implement the event-handler methods. When CgiRemoteControl runs, it invokes the OnPluginControl method.

```
public void Initialize(FTC.SDKInterface.Services.IUnityContainer unityContainer) { }
public void OnSolutionChanged(ISolution solution) { }
public void OnAssetGeneration(string assetFile, bool isTargetAsset, OperationStatus status) {
}
public void OnImportFinished() { }
public void OnShutdown() { }

public int OnPluginControl(string command)
{
    Logger.Info("This is a sample for calling OnPluginControl() method in a plugin by
CgiReomteControl.");
    return 0;
}
```

Specifying the events the plugin handles

Implement the EventsToSubscribe property of the IEventListener interface. This property specifies the set of events to be passed to the plugin. To have CgiRemoteControl call your plugin method, you must include EventId.PluginControlCommand.

```
public EventId EventsToSubscribe => EventId.PluginControlCommand;
```

Below are the sample code and an example of the CgiRemoteControl run results.

```
using Feat.Utils;
using FTC.SDKInterface.Plugins;
using FTC.SDKInterface.Solution;
using System.Collections.Generic;

SampleCgiRemoteControl.cs
namespace SampleCgiRemoteControl
{
    public class SampleCgiRemoteControl : IEventListener
    {
        public EventId EventsToSubscribe => EventId.PluginControlCommand;

        public void Initialize(FTC.SDKInterface.Services.IUnityContainer unityContainer) { }

        public void OnSolutionChanged(ISolution solution) { }

        public void OnAssetGeneration(string assetFile, bool isTargetAsset, OperationStatus
status) { }
```

```
        public void OnImportFinished() { }

        public void OnShutdown() { }

        //This method is called when launching CgiRemoteControls.
        public int OnPluginControl(string command)
        {
            Logger.Info("This is a sample for calling OnPluginControl() method in a plugin by
CgiReomteControl.");
            return 0;
        }

        public string Description
        {
            get { return "A plugin running on the CgiRemoteControls"; }
        }

        public IEnumerable<int> FeatureIds { get { return null; } }

        public string Name
        {
            get { return "Sample CgiRemoteControl plugin"; }
        }

        public string Version
        {
            get { return "1.0.0"; }
        }
    }
}
```