**aci** – The AK command Interpreter

Version 1.0

# Table of Content

[Table of Content 2](#_Toc115998723)

[1 Introduction 3](#_Toc115998724)

[2 Project setup 4](#_Toc115998725)

[3 Document Version History 5](#_Toc115998726)

# Introduction

This documentation describes how a new script can be created.

# Create a new script

## Project setup

### Visual Studio

1. Create a new Dynamic Link Library Project.
2. C/C++ (For both Debug and Release)
   1. Add the additional include directory: $(ACI\_CORE)\include
3. Linker (For both Debug and Release)
   1. Add the additional library directory: $(ACI\_CORE)\x64\$(Configuration);
   2. Add the additional dependency: aciCore.lib
4. Create a new class and derive it from aci::InterpreterObject

|  |
| --- |
| // Note that MYPROJECT\_EXPORT is a macro that will expand to: // \_\_declspec(dllexport) when building the library, and // \_\_declspec(dllimport) when using this library from external sources  class MYPROJECT\_EXPORT MyName : public aci::InterpreterObject {  … |

1. Add the following code to a source file:

|  |
| --- |
| extern "C" {  // This function is called by the interpreter  // In this function we create instances for every interpreter object we  // want to provide  \_\_declspec(dllexport) aci::InterpreterObject\*\* generateObjects(int& \_count)  {  // Create a pointer array that will hold all interpreter objects we  // provide  aci::InterpreterObject \*\* ret = new aci::InterpreterObject \*[1];    // Create a instance for every object and add it to the array  ret[0] = new MyName; // <- We create a new interpreter object and add  // it as the first entry  // Set the count of objects we provide  // (Required so the interpreter knows the length of the array)  \_count = 1;  // Return the array  return ret;  }  } |

# Class descriptions

## aci Core

The aci Core contains all the classes that are needed for processing commands, loading scripts and it also provides Interfaces for the scripts and Interpreter implementations.

### InterpreterObject

The interpreter Object is called a “script” object. It is the entry point for the command to which key associated with it. It provides the output functions to display information on the console.

#### Public functions

|  |  |  |
| --- | --- | --- |
| virtual std::wstring | key(void) const | = 0 |
| virtual bool | handle(const std::wstring&, const std::vector<std::wstring>&) | = 0 |
| void | showInfo(void) |  |
| void | printDelimiterLine(void) |  |
| void | queueDelimiterLine(void) |  |
| void |  |  |
|  |  |  |
|  |  |  |

# Document Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version Nb. | Changes | Editor | Date |
| 1.0 | Creation of document | Alexander Küster | 29.09.2022 |