Overview

A Module is an exported Flow Graph that can be loaded and called from another Flow Graph at any point during the game session.

Any Flow Graph can be converted to a Module simply by creating a new module in the Flow Graph Editor, and copying the Flow Graph contents to this new module.

The advantages of using Modules:

- Same Flow Graph can be used in multiple levels, but only exist in one location.
- Modules can receive unique input values from their caller, allowing them to be very robust.
- Modules can return unique output values to their callers, allowing them to be used in many different situations.
- Modules can be instanced (default behavior), meaning that multiple copies of the same module can be active simultaneously, but running with different inputs

Module Types

There are currently two supported module types.

- **Global**: Modules which are used in multiple levels should be created as Global modules.
- **Level**: Modules only used in a specific level should be created as Level modules.

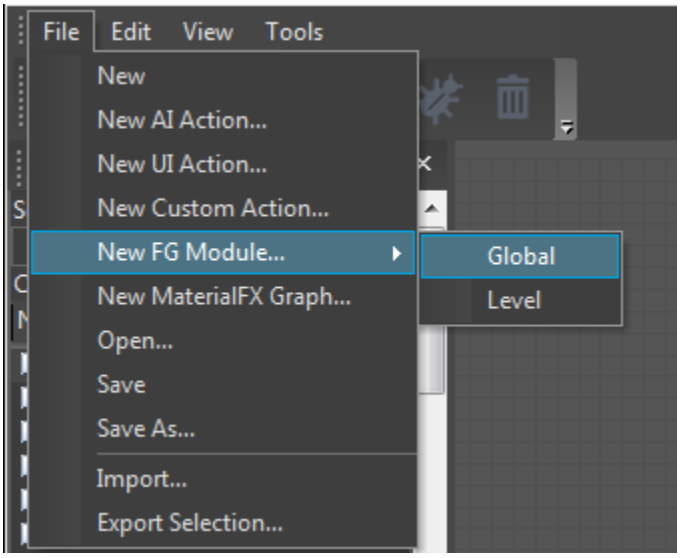
Modules are saved in either of these two locations:

- **Global**: *../GameSDK/Libs/FlowgraphModules*
- **Level**: *../GameSDK/Levels/<levelfolder>/FlowgraphModules*

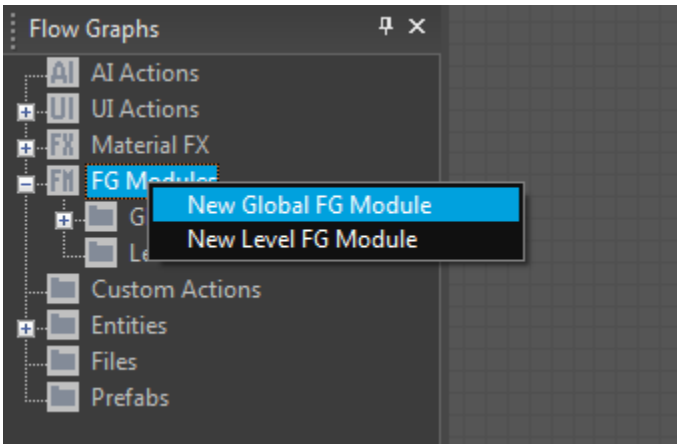
Converting a Flow Graph to a Module

A Module is just a special flavor of a Flow Graph. This means any Flow Graph can become a module with a few simple modifications made to it.

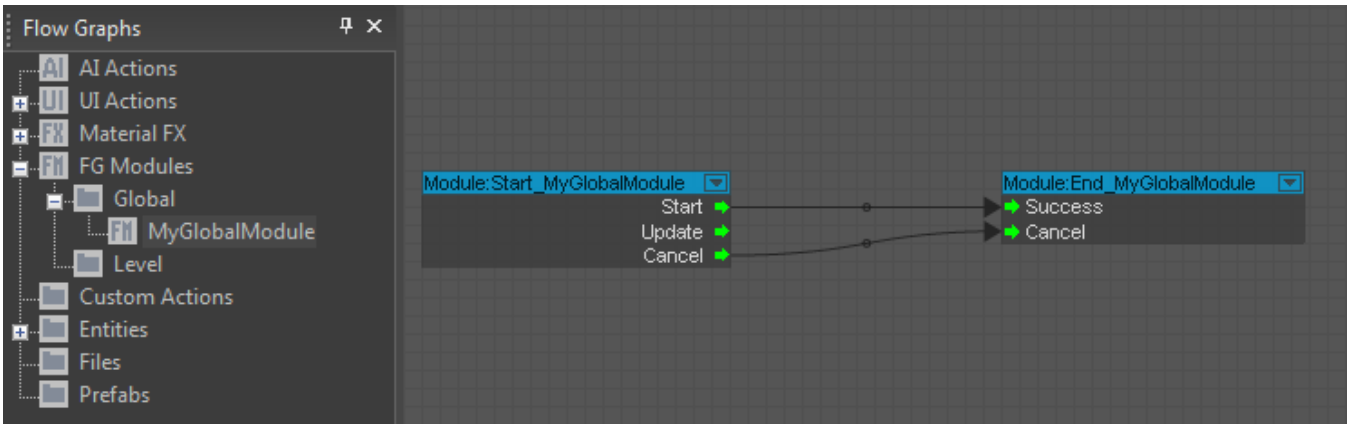
First create a module by selecting **File -> New FG Module -> Global/Level...** in the Flow Graph Editor:



or right-click on the **FG Modules** item in the Flow Graph overview and choose **New Global FG Module / New Level FG Module** from the context menu:



Your new module will appear in the Flow Graph overview:



The new module will already contain two nodes: **Module:Start_<YourModuleName>** and **Module:End_<YourModuleName>**. These are specifically created for this module, and cannot be removed.

Start Node Outputs:

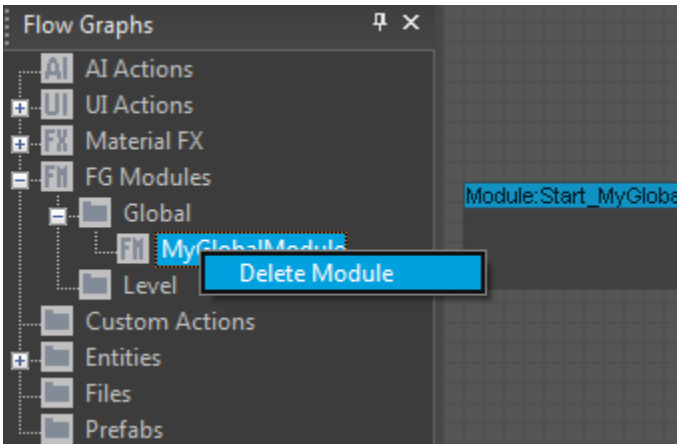
- **Start**: Called when the module was created by the caller node. Treat this like you would the **Game:Start** node.
- **Update**: Called when the caller node wants to pass new parameters or indicate that the internal state of the module should update/reset.
- **Cancel**: Called when the caller node wants to cancel the module execution.

End Node Inputs:

- **Success**: Call this to end the module and pass a '*success*' status back to the caller node.
- **Cancel**: Call this to end the module and pass a '*canceled*' status back to the caller node.

Deleting a Module

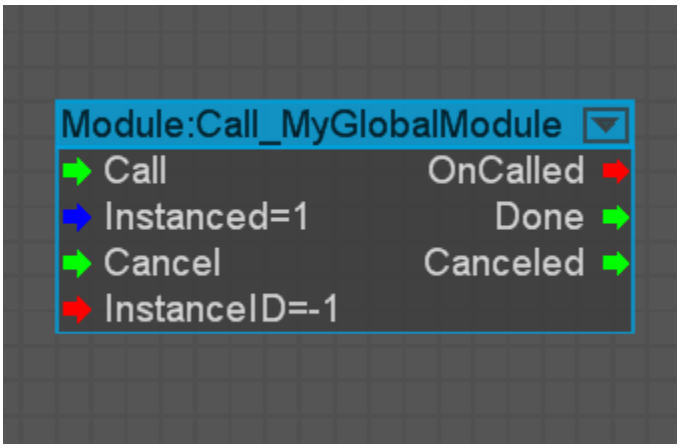
If you want to delete a specific module you can right click on the module name in the **FG Modules** list.



If you delete a module it will also delete the saved XML file (Undo is **not** supported).

Calling a Module from Flow Graph

To call the Module, all you need to do is use the Call node specific to your module. It will be named **Module:Call_<YourModuleName>**:



Inputs:

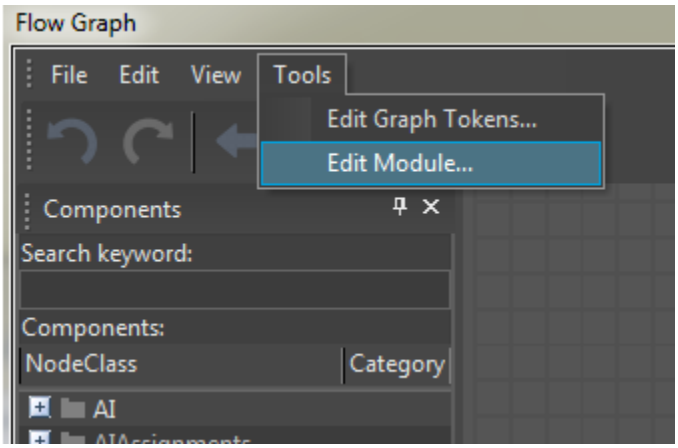
- **Call**: Call to load the module and begin its execution. If the module is already started it will trigger the update port of the Start node with updated parameters (only if not instanced).
- **Instanced**: If set to 1 (default behavior) it will create a new independent instance of the module whenever you trigger the Call input port.
- **Cancel**: Call to cancel this module (needs correct InstanceID if instanced).
- **InstanceID**: InstanceID to identify a module instance. Can be -1 (default behavior) to create a new instance, otherwise it will update the given instance (only if instanced).

Outputs:

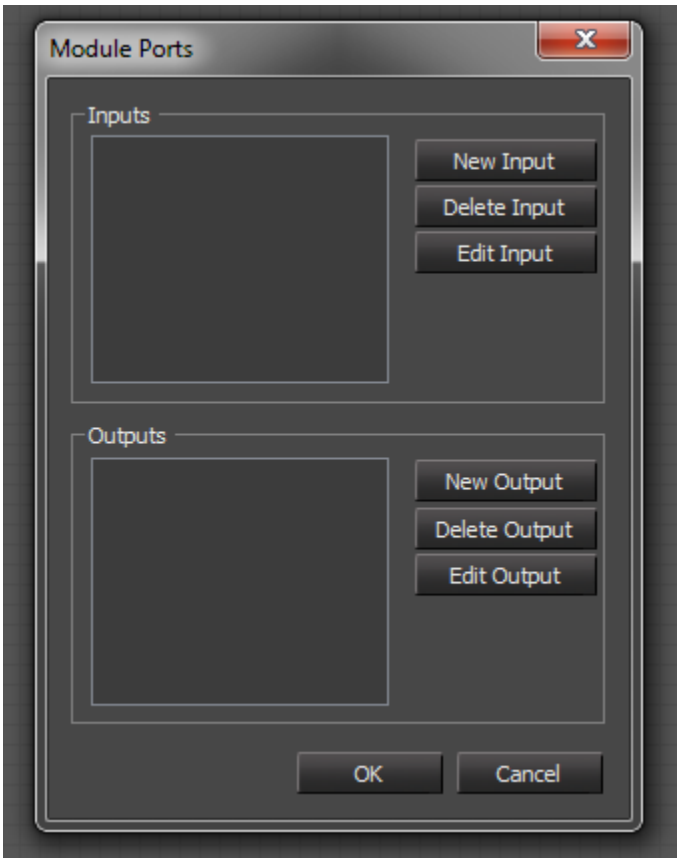
- **OnCalled**: Called when module was started (returns -1 if the module is not instanced).
- **Done**: Called when the module returns with a success status.
- **Canceled**: Called when the module returns with a fail status.

Custom Module Ports

You can also customize the inputs and outputs for each module, to pass extra data back and forth. To do so, select your module and use the **Tools -> Edit Module** menu option in the Flow Graph editor:

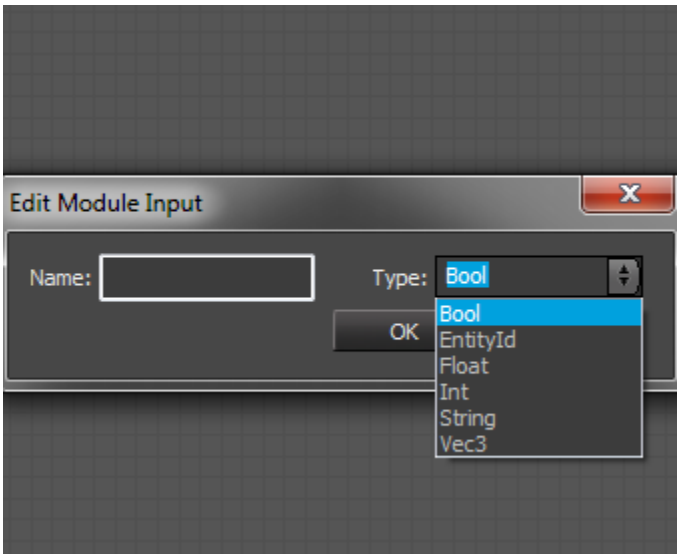


This brings up a dialog which allows adding inputs and outputs:

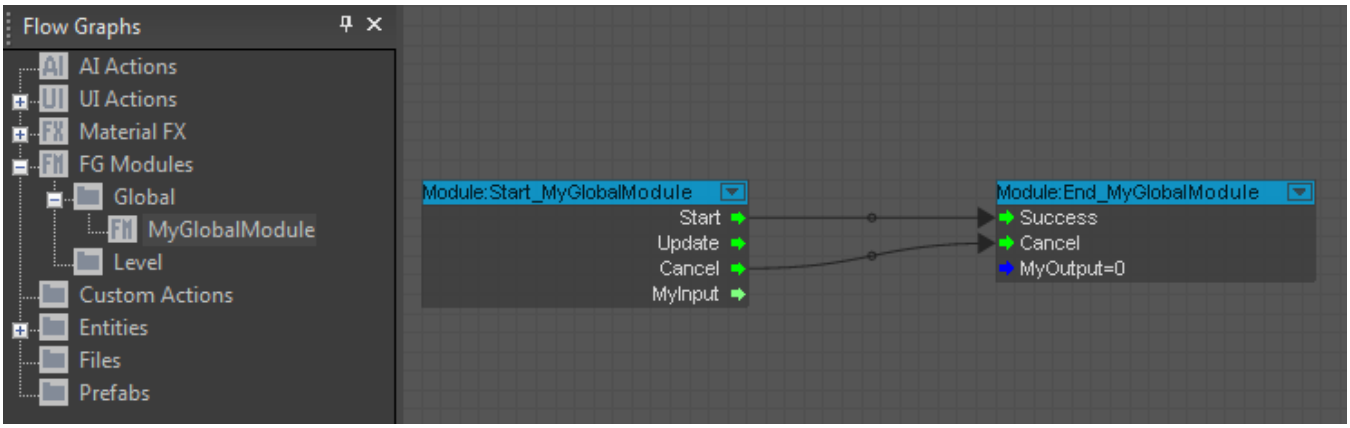


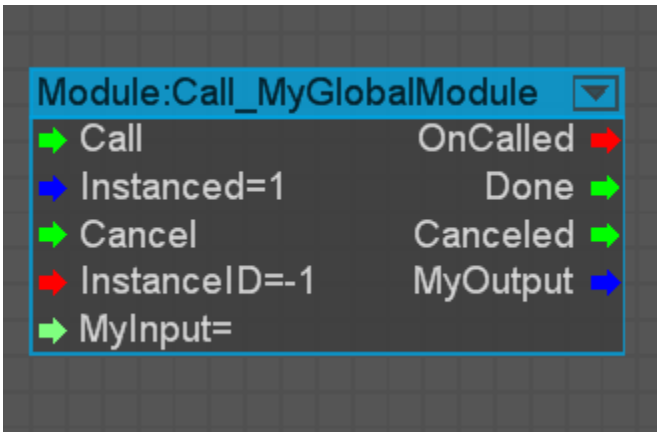
Supported data types are:

- **Bool**
- **EntityId**
- **Int**
- **Float**
- **String**
- **Vec3**



Clicking **OK** on this dialog will regenerate the module nodes (**Start**, **End** and **Call**) with the new inputs and outputs:





All inputs passed to the **Call** node will activate the corresponding outputs on the **Start** node, and similarly inputs to the **End** node will be passed back to the **Call** node when **Success** or **Cancel** are activated.

C++ Interface

To access the Flow Graph Module System from outside you can use the **IFlowgraphModuleManager** interface.

Code/CryEngine/CryCommon/IFlowgraphModuleManager.h

```
#include "IFlowgraphModuleManager.h"
//...
IFlowgraphModuleManager* pModuleManager = gEnv->pFlowSystem->GetIModuleManager();
```

Calling a Module from C++

In some cases it might be needed to start module instances directly from code and pass results back to C++. In this case you can do the following:

Calling a module from code

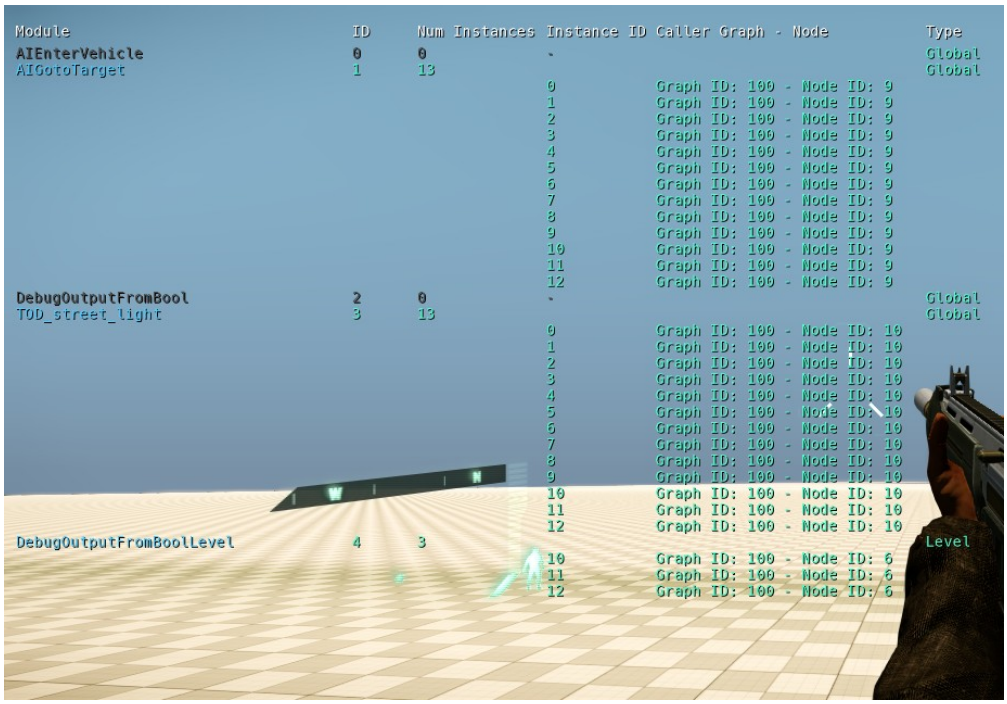
```
void CMyClass::MyModuleCallback(bool bSuccess, const TModuleParams& outputParams)
{
   // do something with the output parameters
}

void CMyClass::CallMyModule()
{
   IFlowGraphModuleManager* pModuleManager = gEnv->pFlowSystem->GetIModuleManager();
   if (const IFlowGraphModule* pModule = pModuleManager->GetModule("MyModule"))
   {
      TModuleParams inputParams;
      //… add input parameters (has to match the inputs in the module)
      pModuleManager->CreateModuleInstance(pModule->GetId(), inputParams, functor(*this, &CMyClass::
MyModuleCallback));
   }
}
```

Debugging a Module

In order to get an overview of all currently available Flow Graph modules and/or see currently active module instances it is possible to use the following console variables:

- **fg_debugmodules** 0/1/2 - 0: Disabled / 1: Shows a list of all currently available modules / 2: Shows a list of all currently available modules and active module instances
- **fg_debugmodules_filter** "filterstring" - "filterstring" can be used to only show specific modules



- **Module**: The name of the module.
- **ID**: Internal module ID.
- **Num Instances**: Number of currently active instances of the module.
- **Instance ID**: Instance specific ID.
- **Caller Graph - Node**: Flow Graph and Node ID of the Flow Graph who called the module.
- **Type**: Global or Level module.

Grayed out modules are modules with no active instances.