IEVENTBLOCK INTERFACE

View online: https://www.construct.net/en/make-games/manuals/addon-sdk/reference/model-interfaces/ieventblock

The `IEventBlock` interface represents an event block in the event sheet. Event blocks are the most important kind of event, and consist of a number of conditions, actions to run when the conditions are met, and optionally further sub-events. It derives from IEventParentRow.

Creating an event block

The following code sample demonstrates the calls necessary to add an *On start of layout* event to the associated event sheet for a given ILayoutView. This is useful with the Custom Importer API, since the AddDragDropFileImportHandler callback provides the ILayoutView that content was dropped in to.

```
// Note: this code is assumed to be in an async function
// First get the associated event sheet for the layout view
const eventSheet = layoutView.GetLayout().GetEventSheet();
if (eventSheet) // note the layout may not have an event sheet
{
        // Get the IObjectType for the System plugin
        const systemType = eventSheet.GetProject().GetSystemType();

        // Create an empty event block at the root level of the event sheet
        const eventBlock = await eventSheet.GetRoot().AddEventBlock();

        // Add an 'On start of layout' condition
        eventBlock.AddCondition(systemType, null, "on-start-of-layout");

        // Example code for adding a 'Set position' action
        //eventBlock.AddAction(iObjectType, null, "set-position", [100, 200]);
}
```

Finding condition and action IDs

Some developer methods are available to explore the list of condition and action IDs that can be used to create events with. See Finding addon IDs for more information.

Methods

AddCondition(iObjectClass, reserved, cndId, params)

AddAction(iObjectClass, reserved, actId, params)

Add a condition or action to this event block. These methods are very similar so they are documented together. *iObjectClass* must be an IObjectClass (i.e. an IObjectType or IFamily) to create the condition and action for. The next parameter is reserved for future use; you must pass `null`. *cndId* or *actId* must be a string specifying the condition or action to create; for example the System *On start of layout* condition ID is `"on-start-of-layout"`. If the condition or action uses any parameters, then *params* must be an array with enough elements for every parameter. Each parameter can be a string, number, boolean or IObjectType. Expression parameters use a string, which can be any valid expression (including calculations like `"1+1"` for number parameters); if you pass a number, it will be converted to a string. IObjectClass can also be passed for object parameters.