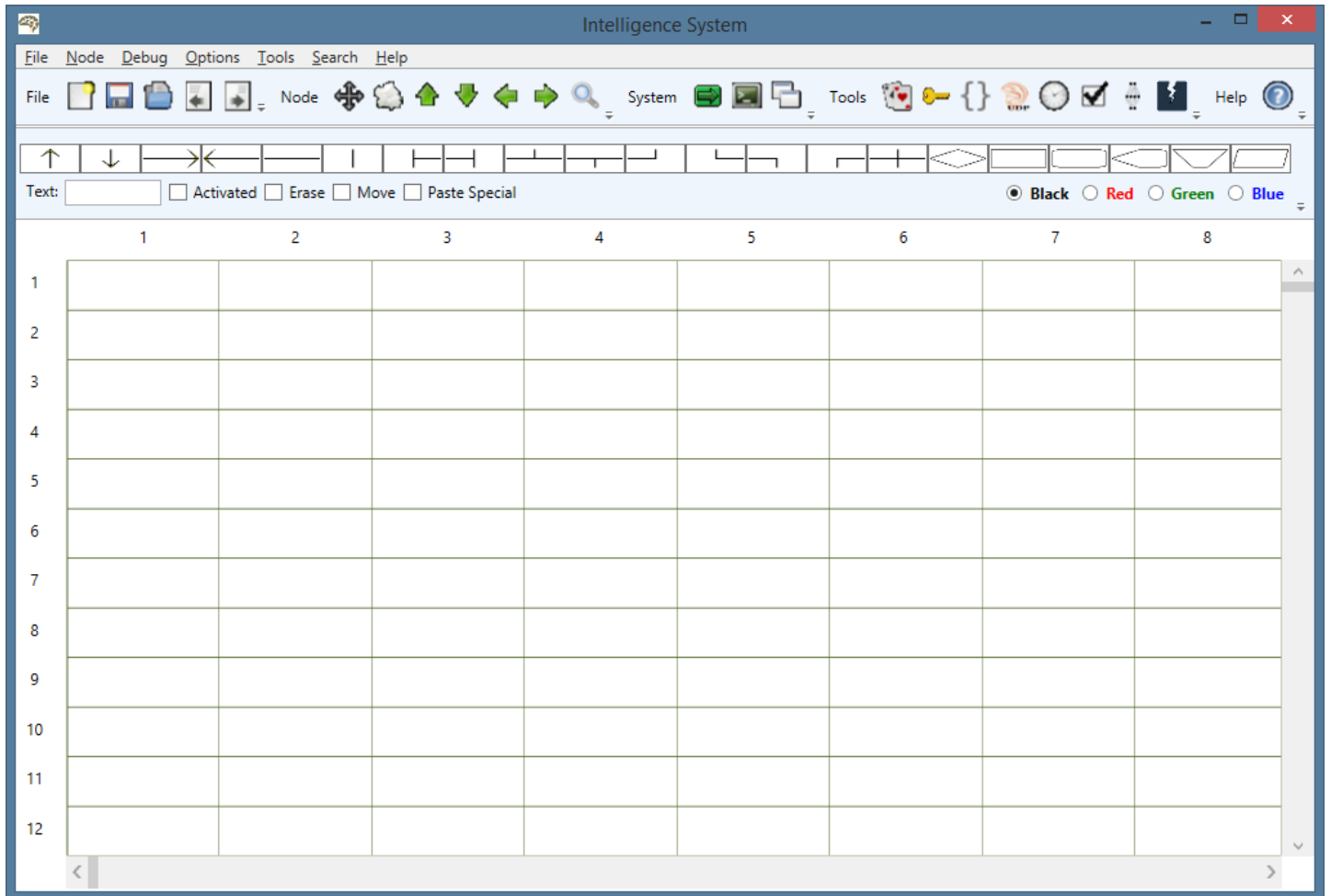


# Intelligence System



Copyright © 2024 IsimSoftware  
Solutions <http://IsimSoftware.com>

# Table of Contents

1	<a href="#">Introduction</a>	1
1.1	<a href="#">How does it operate?</a>	1
2	<a href="#">Creating Flowcharts</a>	2
2.1	<a href="#">Node Properties</a>	2
2.1.1	<a href="#">Display Text</a>	3
2.1.2	<a href="#">Identifier</a>	3
2.1.3	<a href="#">This is a startup node</a>	3
3	<a href="#">Actions</a>	4
3.1	<a href="#">Activation Actions</a>	4
3.2	<a href="#">Response Actions</a>	4
3.2.1	<a href="#">&lt;CATCH&gt;</a>	5
3.2.2	<a href="#">&lt;FINALLY&gt;</a>	5
3.2.3	<a href="#">&lt;TIMEOUT&gt;</a>	5
3.3	<a href="#">Commands</a>	5
3.3.1	<a href="#">Activate Node</a>	5
3.3.2	<a href="#">Activate Node As New Thread</a>	6
3.3.3	<a href="#">Clear Direct Interaction</a>	6
3.3.4	<a href="#">Close Engine</a>	6
3.3.5	<a href="#">Disable Action Log</a>	6
3.3.6	<a href="#">Disable Event Handlers</a>	6
3.3.7	<a href="#">Disable Timer Actions</a>	6
3.3.8	<a href="#">Enable Action Log</a>	6
3.3.9	<a href="#">Enable Event Handlers</a>	7
3.3.10	<a href="#">Enable Timed Actions</a>	7
3.3.11	<a href="#">Get Neural Network Output</a>	7
3.3.12	<a href="#">Halt System</a>	7
3.3.13	<a href="#">Hide Direct Interaction</a>	7
3.3.14	<a href="#">Launch Condition Case</a>	7
3.3.15	<a href="#">Launch Radio Window</a>	8
3.3.16	<a href="#">Launch Regular Casing Window</a>	8
3.3.17	<a href="#">Make Neural Network</a>	8
3.3.18	<a href="#">Open Breadcrumbs</a>	8
3.3.19	<a href="#">Open Button Pad</a>	9
3.3.20	<a href="#">Open Direct Interaction</a>	9
3.3.21	<a href="#">Open File</a>	9
3.3.22	<a href="#">Open Input Box</a>	9
3.3.23	<a href="#">Remove Neural Network</a>	9
3.3.24	<a href="#">Respond</a>	10
3.3.25	<a href="#">Respond for Client</a>	10
3.3.26	<a href="#">Run Script</a>	10
3.3.27	<a href="#">Set Key</a>	10
3.3.27.1	<a href="#">System Keys</a>	11
3.3.27.1.1	<a href="#">AllKeys</a>	11
3.3.27.1.2	<a href="#">AllScripts</a>	11

3.3.27.1.3	<a href="#">AllUDP</a>	11
3.3.27.1.4	<a href="#">ANN</a>	11
3.3.27.1.5	<a href="#">CaseSelections</a>	11
3.3.27.1.6	<a href="#">ConditionCase</a>	12
3.3.27.1.7	<a href="#">LastReturn</a>	12
3.3.27.1.8	<a href="#">LastSystemResponse</a>	12
3.3.27.1.9	<a href="#">LastUserResponse</a>	12
3.3.27.1.10	<a href="#">RadioSelection</a>	12
3.3.28	<a href="#">Show Alert Box</a>	12
3.3.29	<a href="#">Show Yes / No Alert</a>	13
3.3.30	<a href="#">Show Yes / No / Cancel Alert</a>	13
3.3.31	<a href="#">Start UDP</a>	13
3.3.32	<a href="#">Stop UDP</a>	13
3.3.33	<a href="#">Train Neural Network</a>	13
3.3.34	<a href="#">Write UDP</a>	13
4	<a href="#">Direct Interaction</a>	14
4.1	<a href="#">Direct Interaction Commands</a>	14
4.1.1	<a href="#">start node</a>	14
4.1.2	<a href="#">start system</a>	15
4.1.3	<a href="#">clear</a>	15
4.1.4	<a href="#">halt system</a>	15
4.1.5	<a href="#">exit</a>	15
5	<a href="#">Tools</a>	16
5.1	<a href="#">Condition Cases</a>	16
5.2	<a href="#">Event Handlers</a>	17
5.3	<a href="#">Neural Networks</a>	18
5.4	<a href="#">Scripting Engine</a>	19
5.5	<a href="#">Timer Actions</a>	20
5.6	<a href="#">UDP Listeners</a>	20
5.7	<a href="#">User Variables</a>	21
5.8	<a href="#">Wildcard Classes</a>	21
6	<a href="#">Options</a>	23
6.1	<a href="#">Activate Activity Log</a>	23
6.2	<a href="#">Activate Autosave</a>	23
6.3	<a href="#">Activate Interaction Log</a>	23
6.4	<a href="#">Activate System Timer</a>	23
6.5	<a href="#">Activate Event Handlers</a>	23
6.6	<a href="#">Button Outlines</a>	24
6.7	<a href="#">Copy / Move Node</a>	24
6.8	<a href="#">Delete Node</a>	24
6.9	<a href="#">Make Flowchart Webpage</a>	25
6.10	<a href="#">Move All Columns Left</a>	25
6.11	<a href="#">Move All Columns Right</a>	25
6.12	<a href="#">Move All Rows Down</a>	25
6.13	<a href="#">Move All Rows Up</a>	25

6.14	<a href="#">Response Speech</a>	26
6.15	<a href="#">Search Nodes</a>	26
6.16	<a href="#">Set Edit Password</a>	26
7	<a href="#">Command Line Options</a>	27
7.1	<a href="#">No Parameters</a>	27
7.2	<a href="#">Opening a Grid Position</a>	27
7.3	<a href="#">Generic Start</a>	27
7.4	<a href="#">With Node Identifier</a>	27
8	<a href="#">File Options</a>	28
8.1	<a href="#">Agent Files</a>	28
8.2	<a href="#">Settings Files</a>	28
8.3	<a href="#">Import / Export Files</a>	28
9	<a href="#">Toolbars</a>	30
9.1	<a href="#">Main Toolbar</a>	30
9.2	<a href="#">Painter Bar</a>	30
10	<a href="#">System Redistribution</a>	32

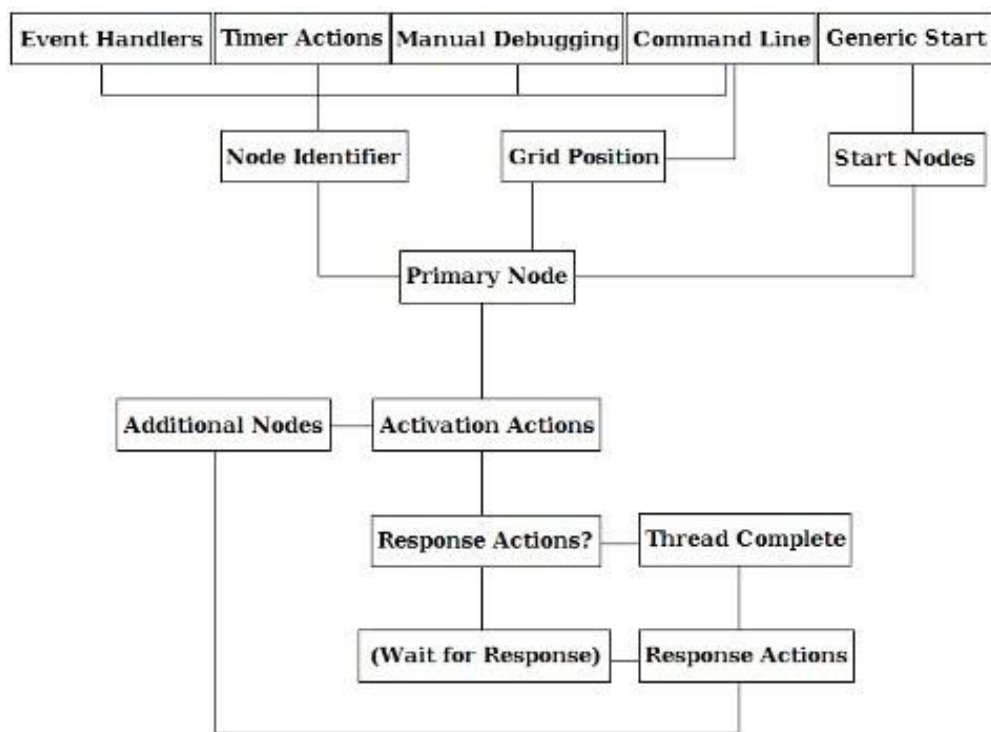
The Intelligence System is a powerhouse engine of creativity that was developed to increase efficiency and save developers time and businesses money. With a visual design, the Intelligence System allows for the simultaneous design and development of flowcharts, diagnostic and troubleshooting systems, wizards, installation packages, call scripts, chat bots, and more. Software engineering features such as built-in recursion, threading, socket writers and listeners, timer actions, voice recognition, voice synthesis, flow control, interactive dialogs, system keys, event handlers, neural networks, and condition casing make .Net software and artificial intelligence development easy.

[Back to Table of Contents](#)

### 1.1 How does it operate?

The Intelligence System achieves its flexibility with nodes. Nodes are base units that contain properties and can perform actions. Every Intelligence System project contains 10,000 nodes (100 rows and 100 columns).

Once activated by starting the system, debugging, or with an event handler or the system timer, a node can activate one or many additional nodes. Every primary (starting) node runs on its own thread, and the Intelligence System can therefore process multiple threads simultaneously. Coupled with a point-and-click visual interface, complicated systems can be created with minimal effort and maximal efficiency. Image 1 illustrates the activation flow of the Intelligence System.



*Image 1: Intelligence System activation flow*

[Back to Table of Contents](#)

## 2 Creating Flowcharts

The first step in the creation of an Intelligence System project is the visual design. The design of a project is accomplished entirely with node properties. The only node properties necessary to design a project are the background image and display text. Nodes can contain background images and display text without containing actions, and designating nodes to have only display properties set (e.g., arrows) is encouraged.

[Back to Table of Contents](#)

### 2.1 Node Properties

Each node of an Intelligence System project is associated with activation and response triggers that can perform numerous actions. One of the available options is node activation. When a node is activated from the primary node, the activated node becomes the primary node, and its events are processed. This sequence continues until no child nodes are activated and events for the primary node are complete.

Once complete, the thread will automatically close if the Direct Interaction window is not visible. If the Direct Interaction window is visible, the thread will close when the client closes the window. Closing threads when no more actions can be taken maintains a minimal memory footprint and frees system resources for additional processes.

Each node has an identity, display text, activation events, and response events; additionally, each node can be declared as a start node. Some nodes are only used to increase readability during the designing phase. Image 2 is a screen capture of the Define Node window that is opened when a button on the main designer is clicked.

The screenshot shows a window titled "Define Node" with a standard Windows-style title bar. Inside the window, there are two input fields at the top: "Identifier" and "Display Text". Below these, the window is divided into two main sections: "On Activation" on the left and "On Response" on the right. Each section contains a table with columns for "Action" (under On Activation) and "Trigger", "Action", and "Value" (under On Response). Below each table are four buttons: "Move Up", "Move Down", "Add", and "Remove Selected". At the bottom of the window, there is a checkbox labeled "This is a startup node" and a large "Apply and Close" button.

*Image 2: Define Node Window*

[Back to Table of Contents](#)

### 2.1.1 Display Text

The display text properties is only used during system design. The content of this property is displayed in the grid button of the associated node. Display text can be used in conjunction with a background image.

[Back to Table of Contents](#)

### 2.1.2 Identifier

The identity property is what identifies the node in the system. While nodes marked as start nodes will activate when the system is started, and therefore do not necessarily need an identifier on the first run, non-start nodes are called in activation and response events using the identifier property. If a start node is returned to at any point during operation, its identifier will be how it is activated, and event handlers, the system timer, and other forms of activation rely on the node identifier.

[Back to Table of Contents](#)

### 2.1.3 This is a startup node

The IsStartNode property is what indicates to the system that a node is the first in the system. Many systems can be ran at the same time, and many nodes can therefore have this property set to true; however, as each primary node runs on a unique thread, there will be a new instance of the Direct Interaction window for each startup node. The system can be started with the command line, the Start System option in the Debug menu in the designer, the F5 key (if the main design window is visible) and by typing "start system" (without quotes) in the Direct Interaction window.

[Back to Table of Contents](#)

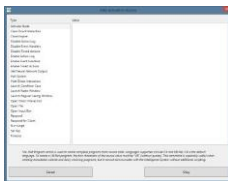
### 3 Actions

The Intelligence System is event-driven. Processing is initiated when an event of some sort occurs. Events in the Intelligence System include starting the system (which activates primary nodes), activating a node manually with the identifier, Node Activation events, Response Events, Event Handlers (when sets of conditions are true), and Timer Actions.

[Back to Table of Contents](#)

#### 3.1 Activation Actions

Activation actions process when a node is activated. The standard life-cycle of a node is: **Activation Actions** → **Wait for response** → **Response Actions** → **Close Thread**. If the activation commands include "C# Program", "Open File", or "Activate Node", the standard interactive process flow is superseded. When a C# Program is ran or file is opened, the process flow continues, but the file or program opens in a new thread. Activating a node reroutes the system immediately using the activated node. Image 3 is a screen capture of the Activation command window.



*Image 3: Activation Command Creation window*

[Back to Table of Contents](#)

#### 3.2 Response Actions

Response actions are processed after Activation actions are processed and a response is received from the user in the Direct Interaction window. Each response command consists of a command, similar to activation commands, and a response trigger. Response triggers are always in lowercase. When what the user types matches the response trigger, the associated commands are processed.

Response triggers can contain wild cards, which are represented with "\*" (without quotes). For instance, the response "1 2 3 4" would activate the response trigger "1 \* 3 4". Image 4 is a screen capture of the Response command creation and edit window.



*Image 4: Response Command Creation Window*

[Back to Table of Contents](#)



### 3.2.1 <CATCH>

The "<CATCH>" (without quotes) response trigger is used to create actions that activate on the event that the user does not enter text that matches any response triggers. Actions associated with the <CATCH> event usually redirect or halt the system.

[Back to Table of Contents](#)

### 3.2.2 <FINALLY>

The "<FINALLY>" (without quotes) response trigger will always activate. Regardless of whether a response match is found or the <CATCH> trigger activates, actions associated with the <FINALLY> trigger will process. The only exceptions to <FINALLY> actions is if the system is halted or redirected to another node.

[Back to Table of Content](#)

### 3.2.3 <TIMEOUT>

The "<TIMEOUT>" (without quotes) response trigger will wait for a user response for a designated duration. If a response is not received in the allotted time (measured in seconds), the associated node will activate. To use the <TIMEOUT> trigger, the Timeout action is selected and the Response Trigger is defined as "<TIMEOUT NumberOfSeconds TargetNode>" (without quotes).

[Back to Table of Contents](#)

## 3.3 Commands

Node commands contain all of the processing in projects created with the Intelligence System. Some commands relate to system navigation, some pertain to the Intelligence System and User Interaction, and others are useful primarily when used in a script. All commands can be used in conjunction with one another.

[Back to Table of Contents](#)

### 3.3.1 Activate Node

The Activate Node command is how the system is navigated. When the proper event occurs, the Activate Node command can be used to activate the node with the specified identifier. Node activation is immediate, and all actions in the queue are removed when the primary node of a system is changed

[Back to Table of Contents](#)

### 3.3.2 Activate Node as New Thread

The Activate Node as New Thread action will activate a node using a new thread.

[Back to Table of Contents](#)

### 3.3.3 Clear Direct Interaction

The Clear Direct Interaction action will clear text in the Direct Interaction conversation text box. No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.4 Close Engine

The Close Engine action will close the Intelligence System engine entirely (not merely halt the current system). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.5 Disable Action Log

The Disable Action Log action will disable the action log if it is enable (it will do nothing if the action log is not enabled). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.6 Disable Event Handlers

The Disable Event Handlers action will disable all Event Handlers (it will do nothing if Event Handlers are not enabled). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.7 Disable Timer Actions

The Disable Timed Actions action will disable system Timed Action (it will do nothing if Timed Actions are not enabled). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.8 Enable Action Log

The Enable Action Log action will enable the system Action Log (it will do nothing if the Action Log is already enabled). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.9 Enable Event Handlers

The Enable Event Handlers action will enable system Event Handlers (it will do nothing if Event Handlers are already enabled). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.10 Enable Timed Actions

The Enable Timed Actions action will enable system Timed Action (it will do nothing if Timed Actions are already enabled). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.11 Get Neural Network Output

The Get Neural Network Output action can be used to process inputs with neural networks created in the Neural Network Engine. Outputs are saved to the "ANN" (without quotes) system key and can be retrieved with the @GetVar(ANN) option. Output is of the format "{output1, output2, ..n}" (without quotes). The value content for the Get Neural Network Output action is "Identifier input1,input2,..n" (without quotes). Inputs contain no spaces.

[Back to Table of Contents](#)

### 3.3.12 Halt System

The Halt System action will halt processing on (only) the active thread immediately.

[Back to Table of Contents](#)

### 3.3.13 Hide Direct Interaction

The Hide Direct Interaction action will hide the Direct Interaction window (it will do nothing if the Direct Interaction window is not visible). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.14 Launch Condition Case

Condition cases created with the Condition Casing engine can be activated with the Launch Condition Case system action. The only input value needed is the identifier of the condition case. When launched, a checkbox window will open. Matching cases are saved to the ConditionCase system key.

[Back to Table of Contents](#)

### 3.3.15 Launch Radio Window

The Launch Radio Window action will open a window with radio selections and record the text value of the selected radio to the RadioSelection system key. The value content of this action is the radio values (one per line).

[Back to Table of Contents](#)

### 3.3.16 Launch Regular Casing Window

The Launch Regular Casing Window will open a window with checkbox selections and record the newline-delimited selections to the CaseSelections system key. The value content of this action is the checkbox values (one per line).

[Back to Table of Contents](#)

### 3.3.17 Make Neural Network

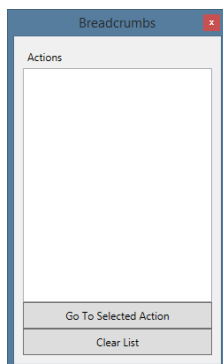
The Make Neural Network action will create a neural network. The value of the action consists of: name inputNeurons hiddenNeurons outputNeurons. For example, to create a neural network with 10 input neurons, 20 hidden neurons, and 30 output neurons, the value would be:

```
myANN 10 20 30
```

[Back to Table of Contents](#)

### 3.3.18 Open Breadcrumbs

The Breadcrumbs window displays recent nodes that have been activated in the system. The list can be cleared, and the user has the option to navigate to a certain node. This action is particularly helpful in conversational systems. Image 5 is a screen capture of the Breadcrumbs window.



*Image 5: Breadcrumbs Window*

[Back to Table of Contents](#)

### 3.3.19 Open Button Pad

The Button Pad is a window containing 0-9 digits. If a system uses numbers to navigate, the button pad can bypass the need to enter numbers manually. When a number in the button pad is clicked, the selected number will be entered into the associated Direct Interaction window. On selection, the one-digit input is sent immediately without the need to hit the return key.

[Back to Table of Contents](#)

### 3.3.20 Open Direct Interaction

The Open Direct Interaction action will show the Direct Interaction window (it will do nothing if the Direct Interaction window is already visible). No value content is needed for this action.

[Back to Table of Contents](#)

### 3.3.21 Open File

The Open File action can run any system command, though it is most commonly used to open files. This action is useful when a node can be enhanced with the presentation of a text file, spreadsheet, or video; additionally, this action is useful when a node runs a batch process. Files are opened with an operating system shell command and therefore open as a new process. Once the file is opened or system command is instantiated, the system will continue without waiting for the file to close or system command to complete.

[Back to Table of Contents](#)

### 3.3.22 Open Input Box

The Open Input Box action will open a prompt and save the user response to the LastUserResponse system key (the same key saved to when a user responds in the Direct Interaction window. This action is a useful way to request input in projects where the Direct Interaction window is not displayed. The value content of the Open Input Box action contains two lines: the first line is what is asked, and the second line is the title displayed on the input box window

[Back to Table of Contents](#)

### 3.3.23 Remove Neural Network

The Remove Neural Network action will remove the neural network with the identifier specified.

[Back to Table of Contents](#)

### 3.3.24 Respond

The Respond command is central to interactive systems that use the Direct Interaction window. When the Respond command processes, the value is appended to the conversation text box in the Direct Interaction window. The Direct Interaction window does not open automatically with this command. The Intelligence System has voice synthesis built-in, and system responses can be spoken by preceding the response text with (MALE) or (FEMALE).

[Back to Table of Contents](#)

### 3.3.25 Respond for Client

The Respond for User command will enter text for the user and send it to the system without prompting. The Respond for User command will activate Response actions, including <CATCH> and <FINALLY> triggers.

[Back to Table of Contents](#)

### 3.3.26 Run Script

The Run Script action will call a script defined with the Scripting Engine tool. If a script is Modal and in C#, it can return a value that is automatically saved to the LastReturn system key (and can be retrieved with @GetVar(LastReturn)). In addition to C# scripts, which are lines of code placed in a predefined string method, the Run Script command can call Entire Program scripts (in C# or VB.Net) in threaded or modal mode; however, only C# scripts can return a (string) value. The value box of the Run Script action should contain the identifier of the script to be ran.

[Back to Table of Contents](#)

### 3.3.27 Set Key

System keys are variables that can be accessed anywhere in system commands. When "@GetVar(myKey)" (without quotes) is entered in the value portion of a command, it is replaced with the value of the key. Keys cannot contain spaces, but key values can contain spaces. All key values are in the string format, and scripts / programs that use keys will have to cast or convert the value. When the Set System Key command is selected, the value is in the format:

myKey this is the value of myKey

An example would be 1), Setting the key titled myKey to "Hello World" (without quotes); then 2), adding a Respond command with the value of "The value of myKey is: @GetVar(myKey)" (without quotes). In the example given, the system would respond "The value of myKey is: Hello World" (without quotes).

In addition to responses, System Keys can be used to inject values directly into C# Script and C# Program code, thereby circumventing the need to pass parameters from the system to the code. The @GetVar feature can also be used to create dynamic node activation and file open values.

[Back to Table of Contents](#)

### 3.3.27.1 System Keys

System keys are one of the backbones of node interactivity. System keys are set with the Set Key action and retrieved with the @GetVar(key) feature. Most system key names are open, but some key names are classified as system keys and written to automatically.

[Back to Table of Contents](#)

#### 3.3.27.1.1 AllKeys

The AllKeys system key will return a comma-delimited string of all system key titles.

[Back to Table of Contents](#)

#### 3.3.27.1.2 AllScripts

The AllScripts system key will return a comma-delimited string of all script titles.

[Back to Table of Contents](#)

#### 3.3.27.1.3 AllUDP

The AllKeys system key will return a comma-delimited string of all system key titles.

[Back to Table of Contents](#)

#### 3.3.27.1.4 ANN

The ANN (Artificial Neural Network) system key is stored to when the neural network engine processes input. The ANN key is in the format "{output1,output2,..n}" (without quotes).

[Back to Table of Contents](#)

#### 3.3.27.1.5 CaseSelections

The CaseSelections system key is written when the Launch Regular Casing action is processed. The CaseSelections key contains the text of one checkbox selection per line.

[Back to Table of Contents](#)

#### 3.3.27.1.6 ConditionCase

The ConditionCase system key is written to when the Condition Casing engine processes. The ConditionCase key contains the text value of one solution per line.

[Back to Table of Contents](#)

#### 3.3.27.1.7 LastReturn

The LastReturn system key is written to when the C# Script action is processed. If the script return a string value, the LastReturn key will contain that value. If the script does not return a string value, the LastReturn key will contain an empty string.

[Back to Table of Contents](#)

#### 3.3.27.1.8 LastSystemResponse

The LastSystemResponse system key is written to when the Respond action processes. The LastSystemResponse contains the content of the response. If the @GetVar(key) was used to generate the response, it will not be in the LastSystemResponse key (but the value of the @GetVar(key) will be).

[Back to Table of Contents](#)

#### 3.3.27.1.9 LastUserResponse

The LastUserResponse system key is written to when an input is entered in the Direct Interaction window; additionally, the key is set when the Respond for Client action processes. The LastUserResponse contains the content of the response. If the @GetVar(key) was used to generate the response with the Respond for Client action, it will not be in the LastUserResponse key (but the value of the @GetVar(key) will be).

[Back to Table of Contents](#)

#### 3.3.27.1.10 RadioSelection

The RadioSelection system key is written to when the Launch Radio Window action is processed. The value of the RadioSelection key is the content of the selection.

[Back to Table of Contents](#)

#### 3.3.28 Show Alert Box

The Show Alert Box action will display a message box with the text given in the value. The first line of the action value is the title of the message box, and the second line of the action value is the message to be displayed.

[Back to Table of Contents](#)



### 3.3.29 ShowYes / No Alert

The Show Alert Box action will display a Yes / No message box with the text given in the value. The first line of the action value is the title of the message box, the second line of the action value is the message to be displayed, and the third line of the action value is the key where the string result will be saved.

[Back to Table of Contents](#)

### 3.3.30 ShowYes / No / Cancel Alert

The Show Alert Box action will display a Yes / No / Cancel message box with the text given in the value. The first line of the action value is the title of the message box, the second line of the action value is the message to be displayed, and the third line of the action value is the key where the string result will be saved.

[Back to Table of Contents](#)

### 3.3.31 Start UDP

The Start UDP action will start the UDP listener with the identifier matching the value content of the action. Incoming messages will be saved to the key specified in the UDP listener definition (and can be viewed in the UDP Listener Tool).

[Back to Table of Contents](#)

### 3.3.32 Stop UDP

The Stop UDP action will stop the UDP listener with the specified identifier.

[Back to Table of Contents](#)

### 3.3.33 Write UDP

The Write UDP action will send a socket message to the address and port specified in the value content of the action. Messages can be multiple words in length. The syntax of the Write UDP action is: "address port message" (without quotes)

[Back to Table of Contents](#)

### 3.3.34 Train Neural Network

The Train Neural Network action will train a network with the given input-output match. The value is separated with newlines and contains name, input, output, and iterations.

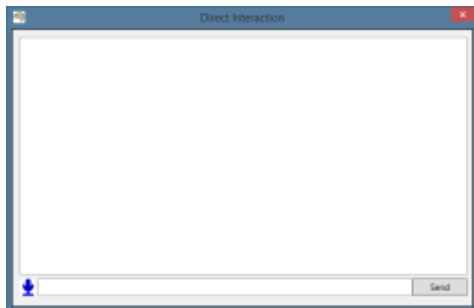
[Back to Table of Contents](#)

## 4 Direct Interaction

The Direct Interaction window is the built-in gateway through which clients interact with Intelligence System projects. The Direct Interaction window can send and receive communications with a project, and Response actions hinge on values being entered into the Direct Interaction window. The button to the left of the response text box toggles the activation of voice recognition.

When used in combination with Wildcard Classes, Timeout autosends, and Voice Synthesis, voice recognition can be used to create systems in which the user speaks to the computer and the computer responds verbally. Depending on the computer running the software, the voice recognition engine may need to be trained before use.

Systems that do not require the direct interaction window can be created, and an action must be created before (after starting the project) the Direct Interaction window will open. The visibility of the Direct Interaction window can be set in every action, and hybrid systems that use the Direct Interaction window in conjunction with other modes of communication are possible. Image 6 is a screen capture of the Direct Interaction Window.



*Image 6: Direct Interaction Window*

[Back to Table of Contents](#)

### 4.1 Direct Interaction Commands

The Direct Interaction window is both a gateway of interaction with projects created with the Intelligence System and an Intelligence System terminal that accepts commands. Commands accepted include start node, start system, clear, halt system, and exit. All system commands are in lowercase.

[Back to Table of Contents](#)

#### 4.1.1 start node

When "start node identity" (without quotes) is entered, where identity is replaced with the node identifier, the system will begin processing with the node identified.

[Back to Table of Contents](#)

#### 4.1.2 start system

When "start system" (without quotes) is entered in the Direct Interaction window, the system will start every node that is set as a starting node will begin processing.

[Back to Table of Contents](#)

#### 4.1.3 clear

When "clear" (without quotes) is entered in the Direct Interaction window, the Direct Interaction Conversation text box will clear.

[Back to Table of Contents](#)

#### 4.1.4 halt system

When "halt system" (without quotes) is entered in the Direct Interaction window, the thread associated with the Direct Interaction window instance will halt processing immediately.

[Back to Table of Contents](#)

#### 4.1.5 exit

When "exit" (without quotes) is entered in the Direct Interaction window, the Direct Interaction window will close. Exiting the Direct Interaction window will automatically halt the thread associated with the Direct Interaction window.

[Back to Table of Contents](#)

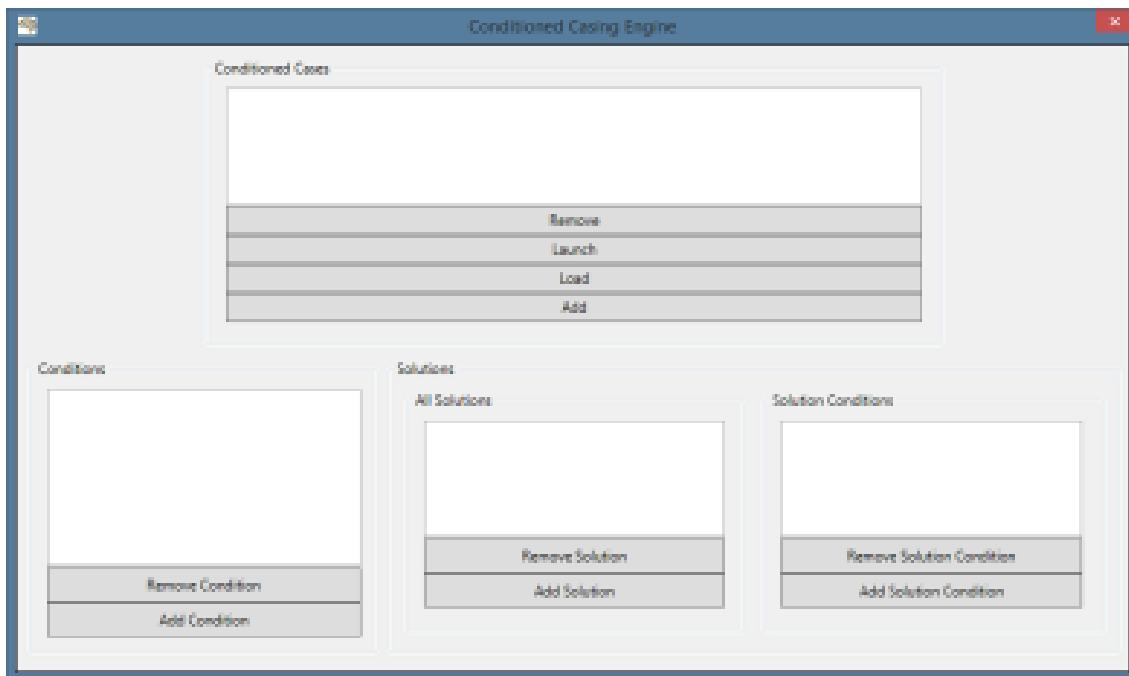
While the base of the Intelligence System is powerful enough to create almost any type of software, there are tools included in the system to make complex development easier. Included in the Intelligence System tools are Condition Cases, Event Handlers, Neural Networks, and Timer Actions.

[Back to Table of Contents](#)

### 5.1 Condition Cases

Condition cases are a form of multi-case positive filtration, and they allow the user to find all solutions that match a given set of conditions (e.g., solutions 2, 4, 6, and 8 all match the conditions "even number" and "less than 10"). Possible solutions contain the answer to be returned and a set of characteristics of that answer that correlate to conditions of the set. The Intelligence System is capable of containing many condition cases.

When the condition case is activated, the window with checkboxes will display. The user will then have the opportunity to select all conditions that are true. When the user closes the window, a result set containing all solutions that can be true will be saved to the system key "ConditionCase" (without quotes). Conditions do not have to be mathematical and can contain diagnostic information (e.g., "car won't turn on", "car won't turn over", and "battery is good"). Image 7 is a screen capture of the Condition Casing Engine.



*Image 7: Condition Casing Engine*

[Back to Table of Contents](#)

## 5.2 Event Handlers

Event Handlers serve as 'when' activators, and they are bound to system keys (that can be set in all actions other than full .Net programs). When activated, the Event Handler engine will repeatedly cycle all condition sets and activate all target nodes every time the condition set is true. Condition sets can contain multiple cases (e.g., "when the system myKey is the value myValue"), and all conditions must be true before the event is handled. As the engine will likely cycle system keys many times per second, it is important for target nodes (that are activated) to reset the system key to a value that will not match the event handler condition set.

To create an event handler, first add the conditions to the upper-right of the window. Next, add target nodes to the lower-right of the window. Once all conditions and target nodes are added, add the Event Handler with a click of the Add Event Handler button to the lower-right of the window.

When an event handler is selected in the Event Handler list, its conditions and targets will load in the lists on the left. A similar event handler can then be created (by clicking the Add Event Handler button) or the selected event handler can be modified (by clicking the Edit Selected Event Handler button). Image 8 is a screen capture of the Event Handler window.

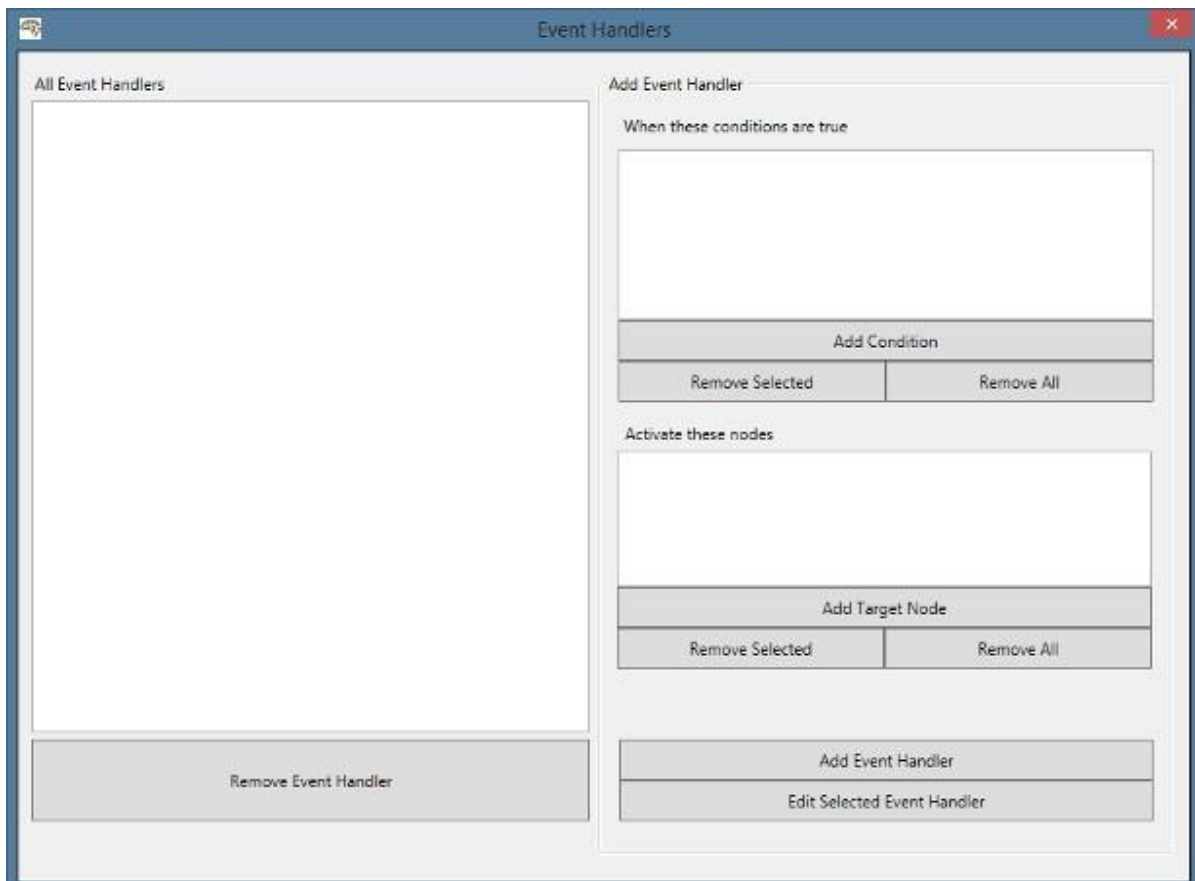


Image 8: Event Handler Engine

[Back to Table of Contents](#)

### 5.3 Neural Networks

Neural networks are useful for pattern recognition and can be added with the Neural Network tool (accessed in the Tools menu). To use neural networks, first define the identifier, input neuron count, hidden neuron count, and output neuron count in the upper-middle groupbox; then, press the 'Make Neural Network' button. Once the network is created, it can be trained.

To train the neural network, enter desired inputs and outputs in the respective textbox. Inputs should be separated with commas and contain no spaces. After an input-output set is defined, click the Add Training set button to add the input-output match. Continue until all desired sets are added.

Next, press the Train Neural Network button to the upper-right. After pressing the Train Neural Network button, a prompt will ask how many times the network should cycle through the training sets. At least 5,000 cycles are recommended. The more cycles that are iterated, the more accurate outputs of the neural network will be.

Once training is completed, Click the Add Loaded Network to List button and save the agent. An already added neural network can be edited by clicking the Edit Selected Neural Network button to the upper-left corner of the window. Editing an existing neural network removes it from the list, and the edited neural network will have to be saved again. Image 9 is a screen capture of the Neural Network Composition window.

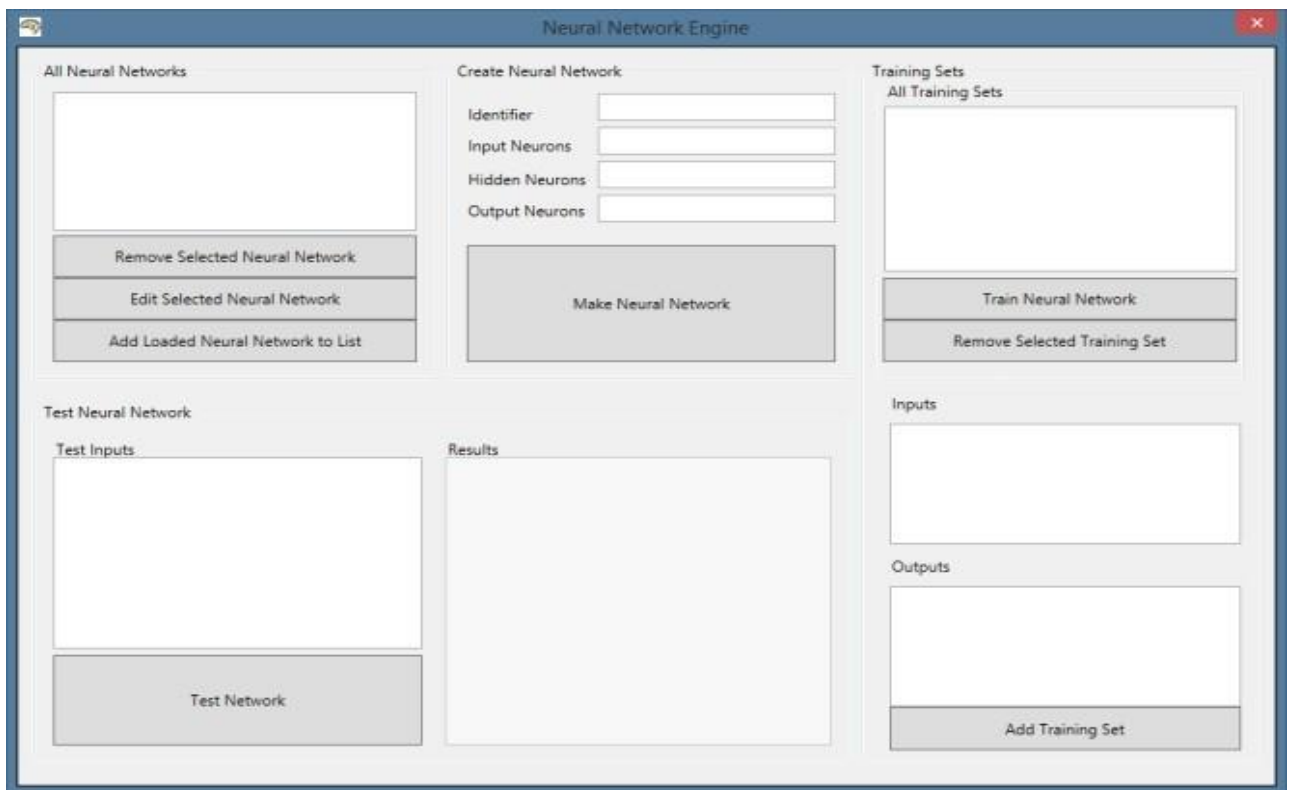


Image 9: Neural Network Engine

[Back to Table of Contents](#)

## 5.6 Scripting Engine

The Scripting Engine provides a centralized location to create scripts and programs. Scripts are bits of code that are placed inside of a method – they can return a string value that is saved to the LastReturn system key, and they can be written in C#. Entire programs contain can be written in VB.Net or C#, but they cannot return a value.

Scripts can be written as Modal (the system thread pauses until the script completes) or threaded (the system continues while the script runs). To return a value, scripts must be modal. The 'Entire Program' option should have a namespace of IntelligenceSystemCode, a class called MainClass, and a method called Main().

While scripts and programs written in the Intelligence System cannot accept parameters, they can have the values of system keys injected directly into the code. To inject the value of a system key into a script, the @GetVar(key) feature is used. An example of GetVar usage is:

```
MessageBox.Show("@GetVar(myKey)");
```

The Scripting Engine can be used to write complex scripts from scratch; however, it is recommended that other tools (e.g., Visual Studio) be used to create the initial scripts (then paste into the Intelligence System). The Intelligence System does not have features such as Intellisense and Syntax Highlighting. If the Intelligence System is the only development environment used, great care is recommended when composing scripts. ***When the script selection changes in the left panel, the script contents (saved or not) are automatically replaced with the content of the script selected in the left panel.*** Image 10 is a screen capture of the Scripting Engine Window.

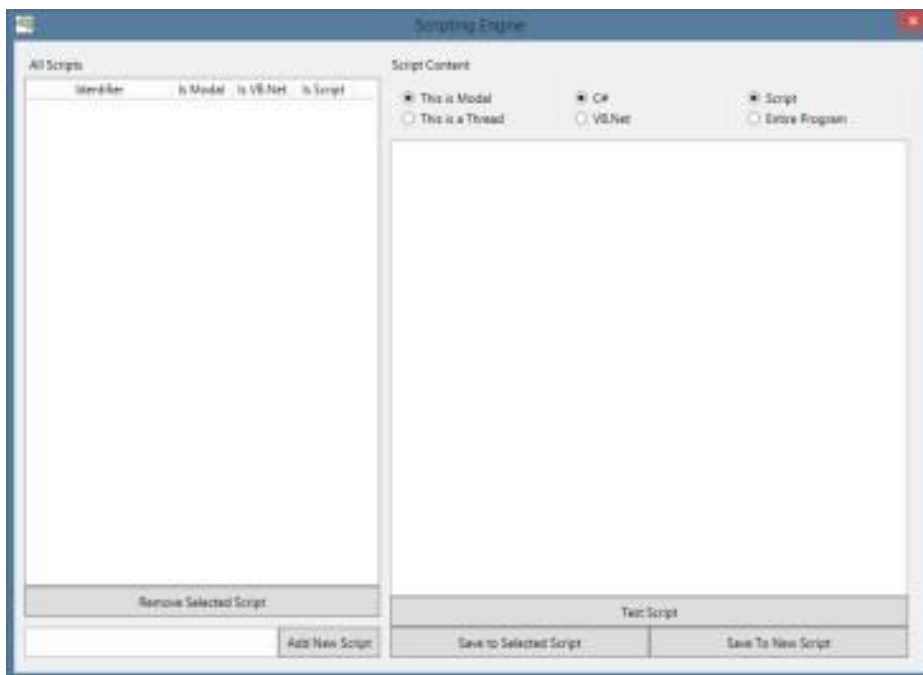
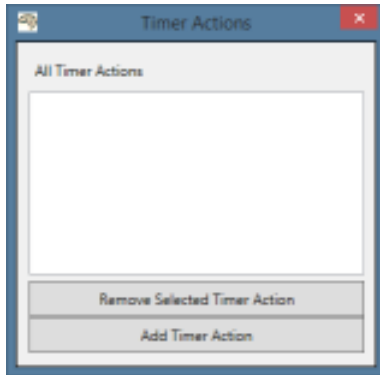


Image 10: Scripting Engine Window

[Back to Table of Contents](#)

## 5.7 Timer Actions

Timer Actions will activate a defined node one time every specified number of seconds. Each activation will open a new thread, and it is recommended that timer actions be used for fast, routine background operations that must be activated at regular intervals. Image 11 is a screen capture of the Timer Actions window.

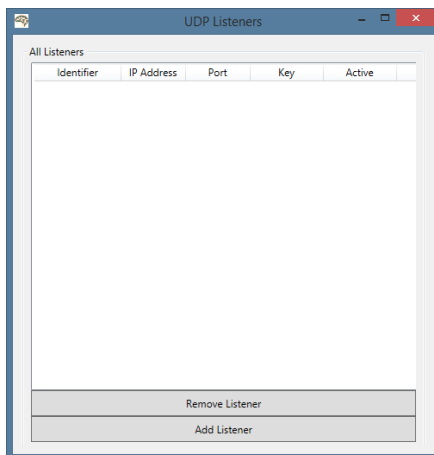


*Image 11: Timer Actions Window*

[Back to Table of Contents](#)

## 5.8 UDP Listeners

The UDP Listeners tool provides a way to create socket listeners that can be started and stopped with node actions. Each UDP Listener contains Identifier, IP Address, Port, Key, and Active properties. The identifier is what is called to activate and deactivate the listener in node actions; the IP Address and Port properties are the locations to be listened to (IP Address can have a value of 'any' [without quotes]); the Key property is the name of the key that will store incoming messages; and the Active property is set to True or False when a Listener is started or stopped. Image 12 is a screen capture of the UDP Listeners window.



*Image 12: UDP Listeners Window*

[Back to Table of Contents](#)



## 5.9 User Variables

The User Variables tool provides a convenient interface to view, add, remove, and modify system keys. System keys are accessed with the @GetVar(key) feature and provide a way for nodes and scripts to communicate. Image 13 is a screen capture of the User Variables window.

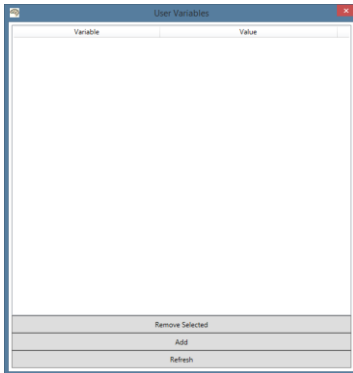


Image 13: User Variables Window

[Back to Table of Contents](#)

## 5.10 Wildcard Classes

Wildcard classes give project designers a way to add custom wildcards to response triggers with c# scripts. Without wildcards, responses are split into words, and each word is then matched with the response trigger to see if the action should run. Wildcards such as \* will match all words:

Response Trigger	1 * 3	
User Response	1 2 3	True
User Response	1 4 3	True
User Response	1 banana 3	True
User Response	3 2 1	False

Wildcard classes can represent words, entire responses, or entered into the response. A wildcard class is a C# script that returns "true" or "false" (with quotes). The word or statement entered by the user is represented in the script with @INPUT:

```
return new List<string>(){
```

```
"banana",  
"apple",  
"orange",  
"tangerine",  
"grape",  
"watermelon"
```

```
}.Contains("@INPUT").ToString().ToLower();
```

The script above can be entered as a word wildcard class called *Fruit* and entered into a Response Trigger such as:

Response Trigger	Is [@Fruit] a fruit	
User Response	Is banana a fruit	True
User Response	Is watermelon a fruit	True
User Response	Is dolphin a fruit	False

Statement wildcards are the same as word wildcards, but the entire statement is sent into the @INPUT keyword. Statement wildcards are useful when custom parsing and input matching is ideal. Since the entire statement is sent into the script, the only word in the response trigger is the name of the script (e.g., {@StatementWildcard}). If a statement wildcard returns "true", actions associated with the response trigger will activate.

Word replacement wildcards actually replace the user response word with the return value of the script. A common use of word replacement scripts is random numbers. No actions activate with word replacements, and additional response triggers with \* wildcards are therefore often added.

[Back to Table of Contents](#)

In addition to tools, there are a number of options available to ease the system development process. Options in the Intelligence System include Activate Activity Log, Activate Autosave, Activate Interaction Log, Activate System Timer, Activate Event Handlers, Button Outlines, Copy/ Move Node, Delete Node, Make Flowchart Webpage, Response Speech, Search Nodes, and Set Edit Password. Additionally, Settings files are covered in the options portion.

[Back to Table of Contents](#)

#### 6.1 Activate Activity Log

The Activate Activity Log option will activate the activity log. The Activity Log will record all actions performed during the operation of a project into a plain text file. Actions and contents are saved to the file (one action per line), and it is therefore recommended that this feature is only used for debugging and maintenance purposes with systems that include source code (which would be visible in the Activity Log).

[Back to Table of Contents](#)

#### 6.2 Activate Autosave

The Auto-Save feature provides a convenient way to save and-or back up the system being developed. When enabled, the user will be prompted for the frequency of each auto-save and file location to save. After a frequency is defined, the system will auto-save at the defined regularity.

[Back to Table of Contents](#)

#### 6.3 Activate Interaction Log

The Interaction Log records all user and system interactions that take place within the Direct Interaction window. Interactions are saved one-per-line to the designated plain text file.

[Back to Table of Contents](#)

#### 6.4 Activate System Timer

The Activate System Timer option turns the System Timer (that activates Timer Actions) on.

[Back to Table of Contents](#)

#### 6.5 Activate Event Handlers

This option will activate the event handler engine. When an event is triggered, the associated nodes will open in a new window

[Back to Table of Contents](#)

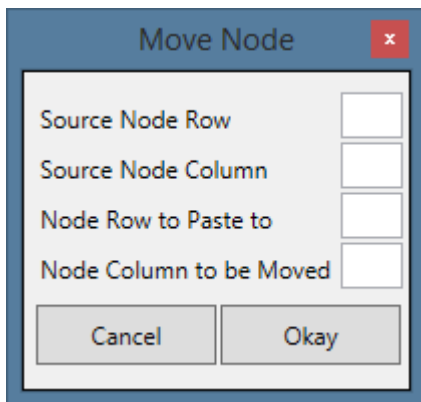
## 6.6 Button Outlines

The button outline feature will toggle button outlines off and on. Not displaying button outlines adds to the aesthetic quality of the system's design. Displaying button outlines assists in defining grid boundaries while designing.

[Back to Table of Contents](#)

## 6.7 Copy / Move Node

Nodes can be copied and moved using options in the Node menu. When the Copy Node window opens, enter the row and column numbers of the source and destination nodes. Image 14 is a screen capture of the Copy Node window.

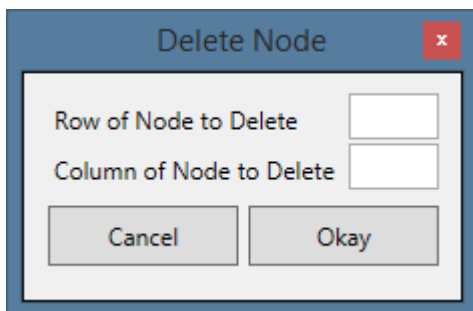


*Image 14: Copy / Move Node Window*

[Back to Table of Contents](#)

## 6.8 Delete Node

Nodes can be deleted using the Delete Node option in the Node menu. When the Delete Node window opens, enter the row and column of the node and click okay. Image 15 is a screen capture of the Delete Node window.



*Image 15: Delete Node Window*

[Back to Table of Contents](#)

## 6.9 Make Flowchart Webpage

The Make Flowchart Webpage option will save the system design to a webpage that can then be viewed or printed.

[Back to Table of Contents](#)

## 6.10 Move All Columns Left

The Move All Columns Left option will move all nodes in the entire system left. The first column(s) will be deleted from the system, and an empty column will be added to the end. This option is useful when maintaining the system design.

[Back to Table of Contents](#)

## 6.11 Move All Columns Right

The Move All Columns Right option will move all nodes in the entire system right. The last column(s) will be deleted from the system, and an empty column will be added to the beginning. This option is useful when maintaining the system design.

[Back to Table of Contents](#)

## 6.12 Move All Rows Down

The Move All Rows Down option will move all nodes in the entire system down. The last row(s) will be deleted from the system, and an empty row will be added to the beginning. This option is useful when maintaining the system design.

[Back to Table of Contents](#)

## 6.13 Move All Rows Up

The Move All Rows Up option will move all nodes in the entire system up. The first row(s) will be deleted from the system, and an empty row will be added to the end.

[Back to Table of Contents](#)

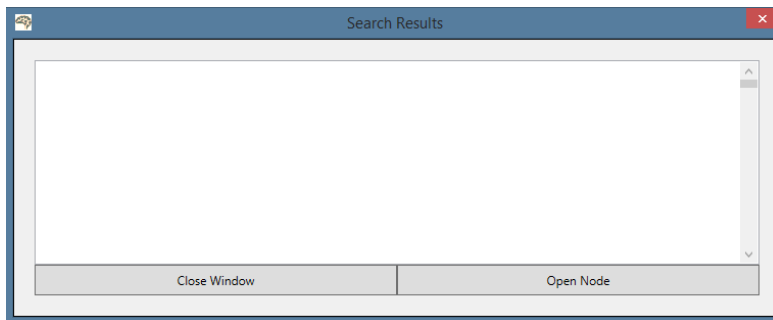
#### 6.14 Response Speech

The Response speech option will cause the system to vocalize the response action with a synthesized voice. Male and female voices can be used for the Response Speech option. The system setting for voice can be overridden in each respond command by prepending the text of the response with "(FEMALE)" (without quotes) or "(MALE)" (without quotes).

[Back to Table of Contents](#)

#### 6.15 Search Nodes

The Search Node feature can be accessed in the Node menu. When the option is selected, the user will be prompted for a search value. After a value is entered, every property of every node in the system will be searched for the phrase. Properties that are found can be selected, and the parent node can be opened, in the Search Node window. Searches are case sensitive. Image 16 is a screen capture of the Search Node window.



*Image 16: Search Nodes Result Window*

[Back to Table of Contents](#)

#### 6.16 Set Edit Password

When a system is complete, the Set Edit Password option can be set to disallow editing while still displaying the system outline. When this feature is enabled, the system can still be edited, but each edit action will require the system password to be entered. There is no way to retrieve edit passwords without opening the Intelligence System in development mode (not available to end users) and recording the edit password is therefore of great importance.

[Back to Table of Contents](#)

## 7 Command Line Options

The Intelligence System can be opened in numerous ways. The designer window can be opened, the system can run with a node identifier, all startup nodes can be processed, or a grid position can be given.

[Back to Table of Contents](#)

### 7.1 No Parameters

When the Intelligence System is opened without any parameters, the designer window will open. This option is the standard way to open during the creation of an Intelligence System Project.

[Back to Table of Contents](#)

### 7.2 Opening a Grid Position

The Intelligence System can start using a grid position if the row and column are given as command line parameters. The Designer Window does not display with this option. The format of the command line call is:

"IntelligenceSystem.exe myfile.isf row column" (without quotes)

[Back to Table of Contents](#)

### 7.3 Generic Start

The Intelligence System can open starting all nodes with the IsStartNode property set to true. The Designer Window does not display with this option. The format of the command line call is "IntelligenceSystem.exe myFile.isf run" (without quotes)

[Back to Table of Contents](#)

### 7.4 With Node Identifier

The Intelligence System can open starting all nodes with a given identifier. The Designer Window does not display with this option. The format of the command line call is "IntelligenceSystem.exe myFile.inf myNode" (without quotes).

[Back to Table of Contents](#)

## 8 File Options

The Intelligence System includes multiple ways to load and integrate systems together. File options for system components include New, Open, Save. In addition to system files, settings can be saved and loaded. The import and export features are the recommended method of loading and saving, and they have access to both settings and system components.

[Back to Table of Contents](#)

### 8.1 Agent Files

Agent files are the primary files of the Intelligence System, and they are loaded and saved with the New, Open, and Save options in the File Menu. Components stored in agent files include the node bank, action definitions, event handlers, timer actions, scripts, neural networks, system keys, and wildcard classes (i.e., everything that is not a setting). The only setting saved to agent files is the Edit Mode Password. Using the New, Open, and Save options will overwrite all components loaded.

[Back to Table of Contents](#)

### 8.2 Setting Files

Setting Files store settings such as the activation values of Event Handlers and Timer Actions, auto-save details, designer settings, and voice-response options. Setting files will overwrite all values in the file, and (if timer settings such as Timer Actions or Event Handlers are activated) timers will start automatically. The use of Setting files is only recommended if the settings are essential to the functionality of the agent.

[Back to Table of Contents](#)

### 8.3 Import / Export Files

The Import and Export File features are the recommended mode of loading and saving agent components. When exporting components, a selection dialog will appear – it is in this dialog that components can be selected for export. Only components selected will be saved.

When importing, the same dialog will appear, and values saved in the file will be pre-selected. Components to be imported can be selected or deselected, and loaded components will not overwrite existing values. Nodes in the import file will only be loaded if the corresponding nodes are currently empty, and collections such as scripts and wildcard classes will be combined with values in the current system.



The import-export feature provides a way for complex systems to be divided among multiple people and re-combine into one agent. As an example, one person can write script A, B, and C; another person can complete node rows 1-10; and another person can compose the wildcard classes. Once all components are completed, the multiple agent files can be combined into one agent by exporting all of the component files and calling the Import feature. Image 17 is a screen capture of the Import-Export window.

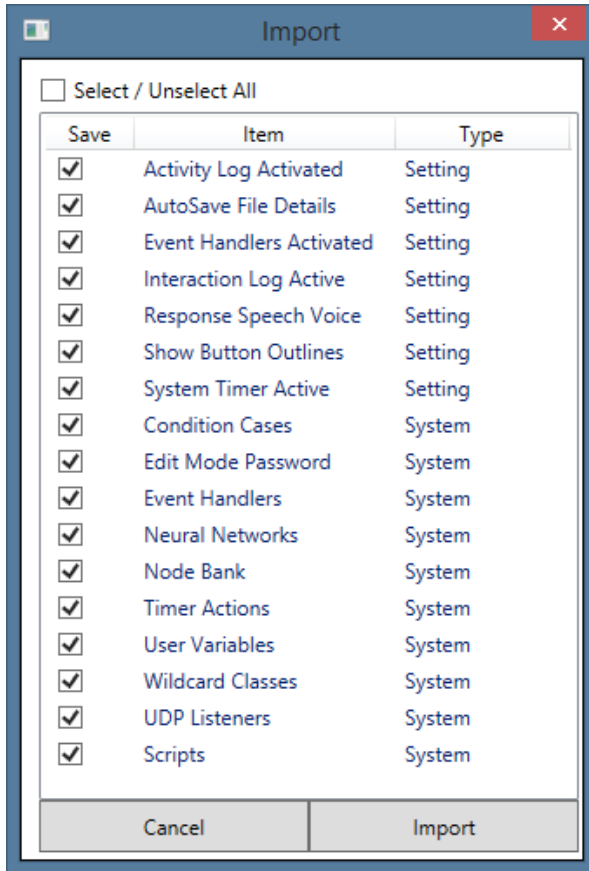


Image 17: Import Window

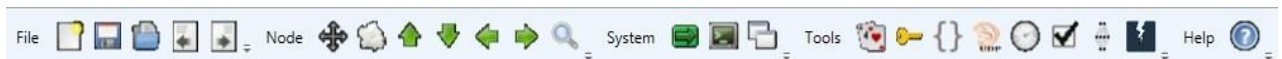
[Back to Table of Contents](#)

The Intelligence System contains two toolbars that provide easy access to common features; these are the Main and Painter toolbars. Toolbar visibilities can be set in the Options menu of the Main Window.

[Back to Table of Contents](#)

### 9.1 Main Toolbar

The Main toolbar contains buttons that allow easy access to file, node, and system abilities; system tools; and the Help file. Tooltips that can be viewed by hovering over each button. Image 18 is a screen capture of the Main toolbar.



*Image 18: Main Toolbar*

[Back to Table of Contents](#)

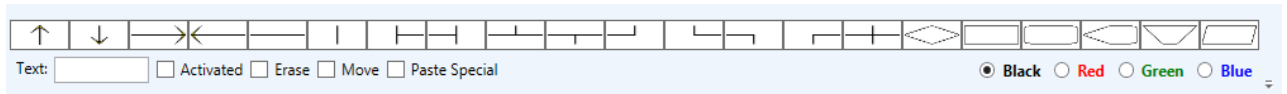
### 9.2 Painter Toolbar

The Painter toolbar contains buttons related to designer features and consists of image buttons, image color selection radios, a display text textbox, and checkboxes for toolbar activation and function definitions. To add an image to a node, click the image first; the clicked image will become highlighted, and the destination node can then be clicked. Text in the textbox will be applied to each clicked node when the image is added. To add only text to an node, uncheck and recheck the Activated checkbox.

Images can be black, red, green, or blue. To choose the color of the image to be applied, select the corresponding radio button to the lower-right of the painter toolbar. To erase nodes, check the erase checkbox, then click the node to be erased.

Nodes can be moved by checking the Move checkbox of the Painter toolbar. Once the Move checkbox is selected, the source node is clicked; then, the destination node is clicked. The above-described method can be repeated.

The Paste Special feature of the Painter toolbar allows for node composition. Once the Paste Special checkbox is checked, the source node is clicked. Every node clicked thereafter will have the contents of the node (display properties and actions) added as long as those values are not already present in the destination node. The Paste Special option can be used to rapidly combine actions without rewriting. Image 19 is a screen capture of the Painter toolbar.



*Image 19: PainterToolbar*

[Back to Table of Contents](#)

Projects created with the Intelligence System can be given or sold to others with the redistributable interpreter. To create a redistributable project, select the Create Redistributable option in the Debug menu. The Redistribute option will create directory of your choosing that contains the redistributable interpreter and a binary project file that cannot be edited with the designer.

Once the project directory is created, the directory can be archived and sold (all files in the directory are necessary). To run the project, start the redistributable interpreter in the same directory as the binary project file. Renaming the redistributable executable is recommended.

[Back to Table of Contents](#)