



视像 JForex

版本 1.6.39

用户指南

Revision 1

10 月 2016

目录

免责声明.....	4
什么是视像 JForex?.....	5
1. 启动视像 JForex 平台	6
1.1. 登录:	6
1.2. 启动并登录到策略处理器:.....	7
2. 菜单, 工作区和工具栏	8
2.1. 菜单	8
2.1.1. File (文件)	8
2.1.2. Compiler (编译器)	8
2.1.3. Help (帮助)	8
2.2. 工具栏	8
2.3. 其它选项	9
3. 起始点	10
4. 模块	11
5. 数据功能库	13
5.1. 技术指标:.....	14
5.2. 策略:.....	15
5.3. 组件	17
5.3.1. 信息:.....	17
5.3.2. 逻辑型	18
5.3.3. 运算	19
5.3.4. 交易	20
5.3.5. 其它	22
6. 变量	24
6.1. 变量面板	25
6.1.1. User's variables (用户的变量) :.....	25
6.1.2. Auto created variables (自动生成的变量) :.....	25
6.1.3. Default variables (默认变量) :.....	25
6.1.4. Account (账户) :	25
6.1.5. Position's info:仓位信息:	25
6.1.6. Trade Event (交易事件) :.....	26

6.1.7.	On Candle（基于蜡烛）:	26
6.1.8.	On Tick（基于 Tick）:	26
6.2.	变量类型:	27
7.	如何	30
7.1.	如何做一个策略的指定	30
7.1.1.	指定交易产品:	30
7.1.2.	指定期限:	30
7.2.	如何使用多个交易产品	31
7.3.	如何处理移动值	33
7.4.	如何使用计数器	35
7.5.	如何在两个指标之间建立交叉	37
7.6.	如何判定和使用仓位	38
7.7.	如何建立 Tick 条形图（通过计数器）	39
7.8.	如何处理逻辑型触发器	42
7.9.	如何缩短数字	43
7.10.	如何使用日期和时间	44
7.11.	如果一个（多个）挂单已成立，如何取消剩余的挂单	45
8.	错误信息和故障排除	46
8.1.	连接信息以及不完善的模块:	46
8.2.	“No Value”（无值）错误	46
8.3.	变量命名:	47
8.4.	变量缺失一个“start value”（起始值）	47
8.5.	高级调试	48

免责声明

本手册，目的是介绍、说明、和解释视像 JForex 的功能和操作。为了迎合交易的高速发展变化，手册内容将尽可能保证及时不断的更新。手册的设计是基于方便用户快速查看某项功能或促进全面系统学习的原则。

虽然手册在尽可能不断的快速更新，但因为流程在不断发展，功能在不断的调整，用户必须充分认识到手册可能不包含所有流程和功能，同样的道理，一些描述和解释可能已经过时，不能反应最新的流程和程序。

因此，强烈建议客户在参考本手册的同时也在模拟账户上测试不同的平台特点和功能。平台的交易环境和功能以及各种周边问题必须由客户和潜在客户审查，所有可能的剩余问题应该在实时交易和管理真实本金前咨询我们公司，并由我们公司给予解答。

我们对损失既不负责也不承担责任，包括但不限于使用本手册导致的交易损失或利润损失，因为手册可能不完整，不准确或者过时。在使用我们公司平台时，本手册不得被认为是贸易咨询、投资咨询、或者作为在杜高斯贝交易平台上是否执行一个具体交易的参考建议。

本手册不涵盖任何我们公司与其客户、潜在客户、或其他合作伙伴之间的商业关系。有关于我们公司对于市场、客户、潜在客户、或其他合作伙伴的操作，我们鼓励读者查看我们公司的网站或联系我们公司的代表。

什么是视像 JForex?

[视像 JForex](#) 是一个全面的创建、测试和运行策略的方案工具；它基于一个友好的用户体验操作界面，使用移动拖放等功能。它拥有一种特殊的方法来，通过模块之间的连接来创建一个完整的、可被执行的外汇策略并以此来构建一套全面的自动交易策略。平台使用的是基于 **Flash** 和 **Java** 语言的网页版操作界面。

视像 JForex 平台



要求:

- 操作系统: Windows, Linux, Apple OS x
- Mozilla Firefox (火狐), Google Chrome (谷歌浏览器), MS explorer (Windows 浏览器) 或者 Safari.
- Java 1.7 和以上版本
- Flash Player

1. 启动视像 JForex 平台

1.1. 登录:



请登录 www.dukascopy.com 并点击“外汇交易”->“视像 JForex-策略制作器”来启动视像 JForex 平台。

大多数浏览器都自带 flash 插件，我们建议使用 Mozilla Firefox（火狐），Google Chrome（谷歌），MS explorer (including Edge) 或者 Safari。

视像JForex - 策略制作器

视像JForex特点

开始使用

[启动视像JForex](#)

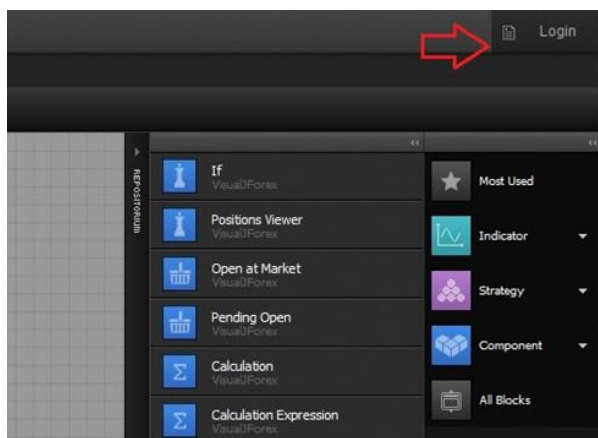
在线助手

视频教程

策略大赛

视像JForex论坛

网络研讨会




当平台启动之后，用户可以选择通过右上角的“Login”使用自己的社区账户进行登录，当然这并不是一个必要的条件。社区账户可以通过该 [网页](#) 创建。

1.2. 启动并登录到策略处理器:

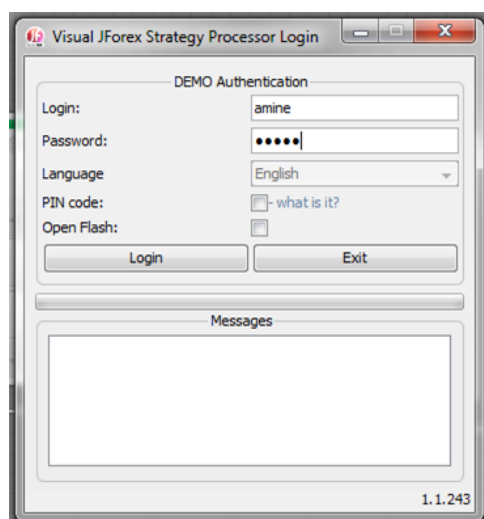
视像 JForex 将被连接到一个交易账户，通过桥接连接平台和交易账户。该桥接被称为视像 JForex 处理器，由 Java 处理连接两个平台并转移必要的用于历史数据测试。

→ 如果没有策略载入或者建立，策略处理器将无法启动。

在建立完策略后，点击“Compiler（编译器）”菜单并选择“Run（运行）”或者点击此按钮来启动策略处理器。在第一次使用的情况下，平台会尝试去连接策略处理器，所以若出现如下信息，显示处理器无法运行的情况是正常的。



点击 Download（下载）来启动策略处理器。一个 Java 文件将被下载并使用 Java 程序执行（Java 网页启动程序），之后将会弹出登录窗口：



输入您的 JForex 模拟账户信息来建立视像 JForex 和 JForex 交易账户之间的连接。

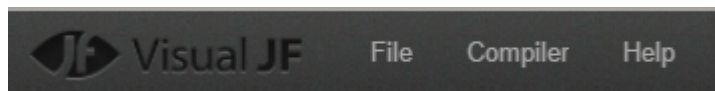
这个连接将允许平台在当前市场数据和历史数据上进行策略的测试。

连接程序只在每次启动时会被建立一次。因此，如果策略处理器被关闭后再次启动，同样的步骤需要重复进行。万一出现连接中断的现象，处理器会自动重新连接，用户会收到警告信息。

或者，处理器也可以从该[链接](#)直接下载。

2. 菜单，工作区和工具栏

2.1. 菜单



2.1.1. File（文件）

通过点击 **File(文件)** > **New(建立新的)**来建立新的策略。为了之后的导入操作，您将被要求填写策略名字。一个以策略名字命名的新的标签将被建立。

通过点击 **File(文件)** > **Open Draft(打开文件)** 来载入已保存的策略文件。用户必须处于连接状态才能载入。点击 **Save Draft(保存)**文件来保存您的策略。

→ 用户可以将视像 JForex 的策略保存在我们的服务器上。

导入和导出功能被应用于保存和载入只有视像 JForex 平台才能使用的拥有特殊格式的文件。这些文件以“.vfs”后缀保存。当然用户也可以选择将文件以 Java 的格式导出。

2.1.2. Compiler（编译器）

编译器菜单将允许把策略放在真实环境或者历史数据上运行。它也将允许建立可以直接在 JForex 平台使用的拥有详尽 Java 代码的策略。可以通过“**View source(查看来源)**”选项来查看此类代码。通过“**Contest(竞赛)**”选项可以建立运行于[杜高斯贝策略竞赛](#)的策略文件。


2.1.3. Help（帮助）


帮助菜单将帮助用户找到我们网上的相关材料或者启动在线客服支持。

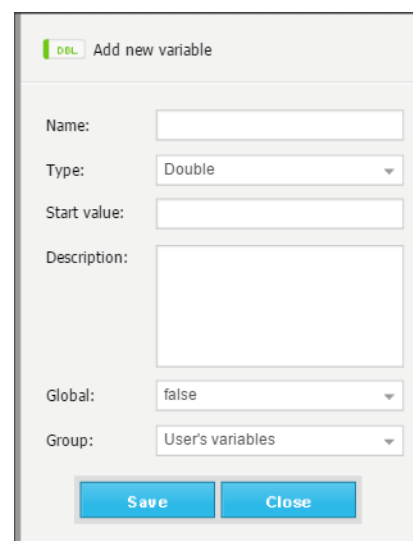
2.2. 工具栏



工具栏提供了一些快捷键、一击保存资料以及运行策略指令。它也有三个额外的功能：


建立新的群组 ；用于建立一个新的群组来管理自定义变量。新的群组将在左边的变量区内以子目录的形式显示。

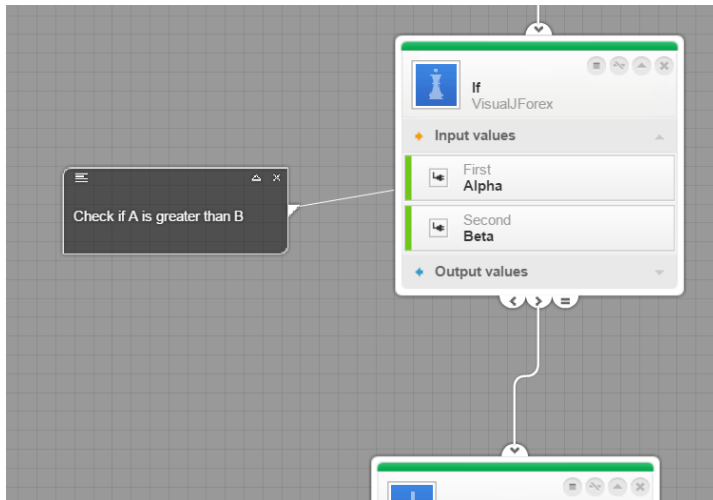
建立新的变量 ；使用这个按钮来建立一个新的变量。一个新的窗口将会弹出，变量的相关参数需要在此设定。（[第 6 部分](#)）



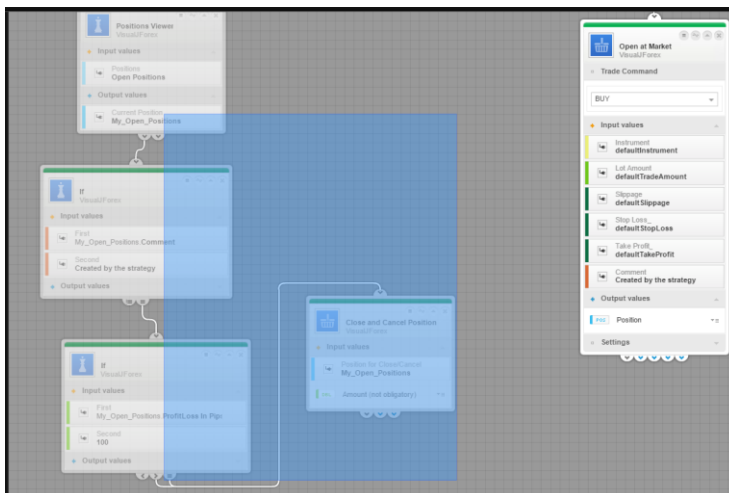
The dialog box titled "Add new variable" contains the following fields and controls:

- Name:** A text input field.
- Type:** A dropdown menu currently set to "Double".
- Start value:** A text input field.
- Description:** A large text area.
- Global:** A dropdown menu currently set to "false".
- Group:** A dropdown menu currently set to "User's variables".
- Buttons:** "Save" and "Close" buttons at the bottom.

插入一个注释 ；点击此按钮来添加一个注释到工作区。注释可以单独存在或者连接到一个模块上。

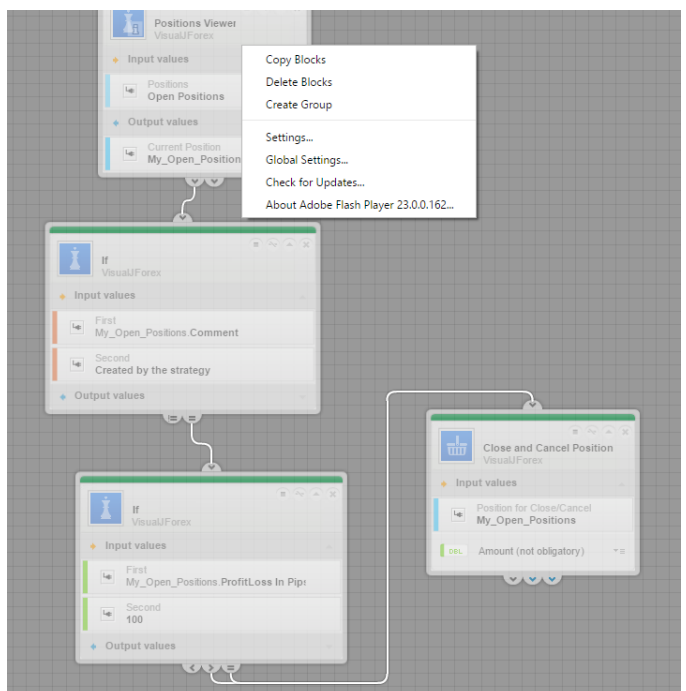


2.3. 其它选项



按住 **Ctrl** 按钮并点击来选择多个模块。在工作区里移动多个选中的模块是可行的。点击键盘上的“Delete（删除）”按钮来删除所选内容。

通过右击显示指令菜单来进行复制、删除多个模块等操作；基于已选中的模块来创建新的群组。



3. 起始点



“起始点”是在平台启动时就显示在工作区的三个模块。他们代表了交易形式和策略的运用。一旦连接了出口箭头，它们就自动被激活了。没有连接的起始点将保持非激活状态，因此其相关策略就不会被应用。这些模块无法从工作区里移除。

→ 一种策略根据其逻辑和要求，可以被应用到一个、两个或三个起始点上。

如上所述，一共有 3 种起始点: On Candle (基于蜡烛), On Tick(基于 Tick) 和 Trade Event(基于交易信息)。

- **On Candle**（基于蜡烛）：这个方案是基于蜡烛时段运行的，它可以自动处理的时间段为：10 秒、1 分钟、5 分钟、10 分钟、15 分钟、30 分钟、1 小时、4 小时、1 天、1 周和 1 个月。任何列出的时间段都可以从“Default Period”（默认时间段）变量的下拉菜单里选择。（[更多信息请参考段落 6.2](#)）
基于蜡烛的起始点被设计用于动态处理上述列出的时间段并且它在每个阶段结束后会由一个闪烁的红灯激活。换言之，当没有特定的时间段被用户应用时，在默认状态下，该起始点将适用于所有时间段并且每 10 秒会闪烁一次。（此为默认情况下最小的蜡烛时间段。）
在平台左边部分的“On Candle”里可以找到属于此起始点的变量数据。（[段落 6.1.7](#)）因此，当起始点闪烁时，它们将获得动态的数值。
- **On Tick**（基于 Tick）：这个方案基于特定交易产品的 tick 价格。默认情况下，基于 tick 方案是适用于 EUR/USD 货币对的。当然，这个也可以在 Default variables（默认变量）部分里进行修改。（[段落 6.1.3](#)）
和基于蜡烛类似，在平台左边的那些基于 Tick 的变量数据是和 tick 价格相关联的。当基于 Tick 起始点被激活后（红灯闪烁），这些数据将被自动应用。
- **Trade Event**（基于交易信息）：这个方案是基于的是平台信息栏里所收到的信息。每当接收到一个特定的信息后就会被激活。

4. 模块

模块被用来建立逻辑型功能或者插入一些现成的技术指标以及交易指令。每一个模块都是独一无二的，各自拥有自己的功能例如执行一个逻辑型对比，处理一个交易指令或者加载一个技术指标。



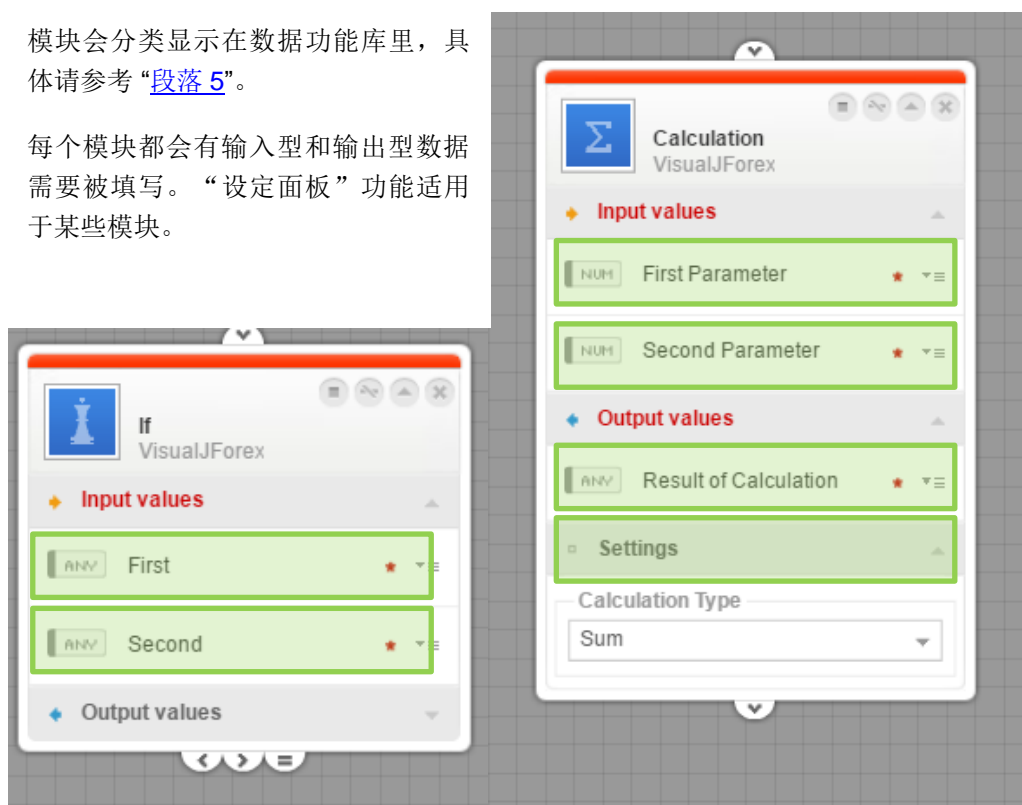
The screenshot shows the 'Open at Market' module interface. It includes a title bar with a green status bar, a toolbar with icons for break point, disconnect, minimize, and remove, and a main area with sections for Trade Command, Input values, Output values, and Settings. Annotations with arrows point to various parts of the interface:

- 进入点:** 这是模块进行连接的进入点。
- 状态栏:** 绿色表示可以被执行；红色表示输入或者输出型变量需要填写完整或者被创建。
- 工具栏 (从左至右):**
 - Break point (分离点):** 此功能可运用于当系统抵达这个模块时中断策略的运行。其主要应用于由于存在问题需要中断策略以及核实功能和变量值。策略可以在之后重新开始。
 - Disconnect block (取消连接模块):** 一击取消所有已连接的模块。
 - Minimize (最小化):** 缩小模块大小来合理安排工作区空间。
 - Remove (移除):** 永久性从工作区移除某个模块。
- 模块的名字**
- 交易指令:** 此内容一般适用于此类交易指令模块；它列出了可适用的订单类型。
- 输入型变量:** 用来满足模块执行的输入型变量。
- 快速进入变量区:** 快速进入相关领域的可用变量。
- 输出型变量:** 自动生成或手动建立的输出型变量。
- 设定:** 模块参数列表。
- 输出点:** 这是输出链接建立的地方。每个模块可以有一个或多个输出链接。其必须被连接才能完全运作。
蓝色输出点和基于交易信息起始点一样是关于订单状态信息的。灰色输出点是基于逻辑型策略来连接其它模块的，因此，其不会受到其它订单状态和确认型信息的影响。

➔ 为了完全运作，任何模块都必须将其输入点和输出点和其它模块互相连接，否则系统将无法识别到此类模块的指令。

模块会分类显示在数据功能库里，具体请参考“段落 5”。

每个模块都会有输入型和输出型数据需要被填写。“设定面板”功能适用于某些模块。



为了执行模块的条件/运行，输入型数据参数必须被填写。默认情况下，输出型数据也是一个必要的变量，其也可以被手动创建。



当模块通过输出点和其它模块相连接后便会被激活并且系统将通过它来执行被下达的指令。

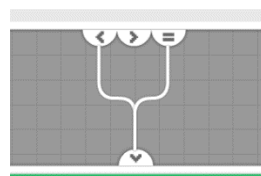
输出点 1: 第一输入变量小于第二输入变量。

输出点 2: 第一输入变量大于第二输入变量。

输出点 3: 第一输入变量等于第二输入变量。

当鼠标放到特定的输出点或者模块范围上的时候会有相关介绍信息显示。

➔ 我们也可以同时连接多个模块的输出点。



5. 数据功能库

数据功能库在平台右侧显示；它是专用于功能和组件的移动型面板。这些项目以标志显示并且可以被拖动到工作区的模块里。数据功能库被分为三大类别：

- **Indicator (技术指标)** - 按照类型列出的技术指标。
- **Strategy (策略)** - 包括在“我的策略”里面的用户的策略。
- **Component (组件)** - 所有按照类别列出的逻辑型模块。

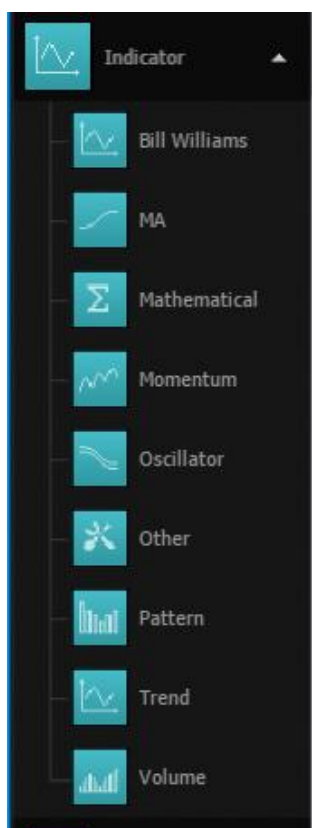


整个部分可以通过红色圈出的按钮收起。它也可以通过橘黄色圈出的按钮来最小化。

指标部分包括已经组建好的技术指标，其可以很简单的被拖动到工作区。

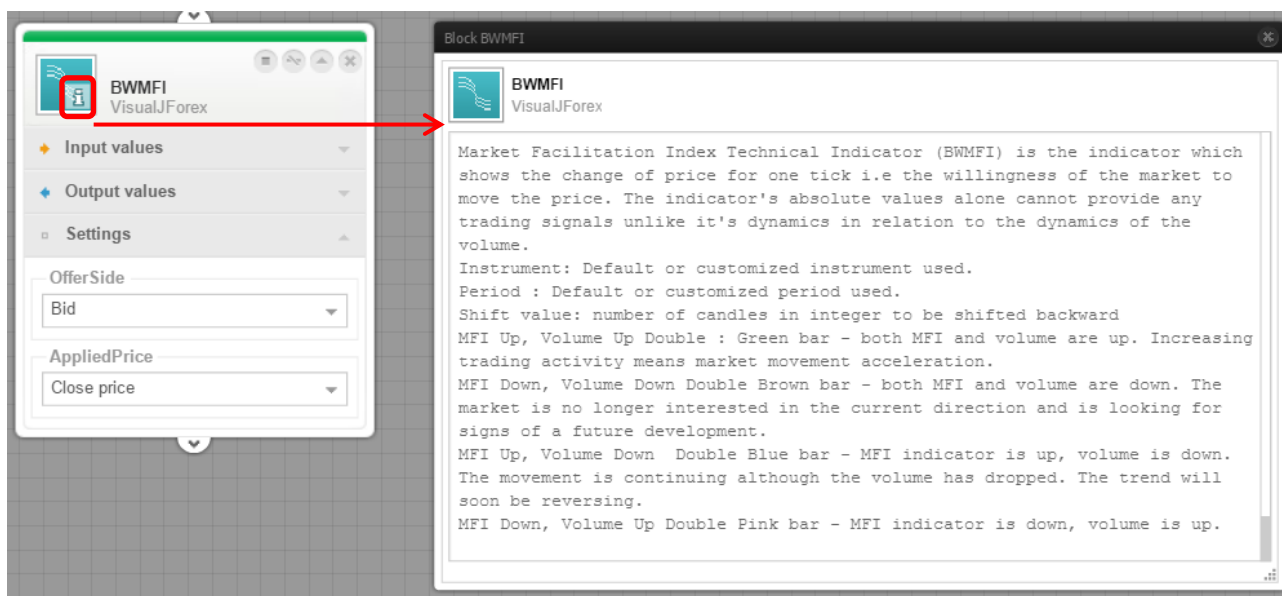
点击导航按钮（黄色圈出部分）来显示工作区预览。对于确认模块位置和策略部分尤其是那些复杂的需要大量模块的策略而言，它是一个非常好的工具。变焦和规模（绿色圈出）按钮可以帮助变焦到 100%。变大和变小是通过鼠标滚轴来实现的，当然也可以通过拖动规模按钮。

5.1. 技术指标:



视像 JForex 提供了非常多的技术指标, 这些指标被分为 9 大类: “Bill Williams”, “Moving Average (MA)”, “Mathematical”, “Momentum”, “Oscillator”, “Pattern”, “Trend”, “Volume” and “Other”.

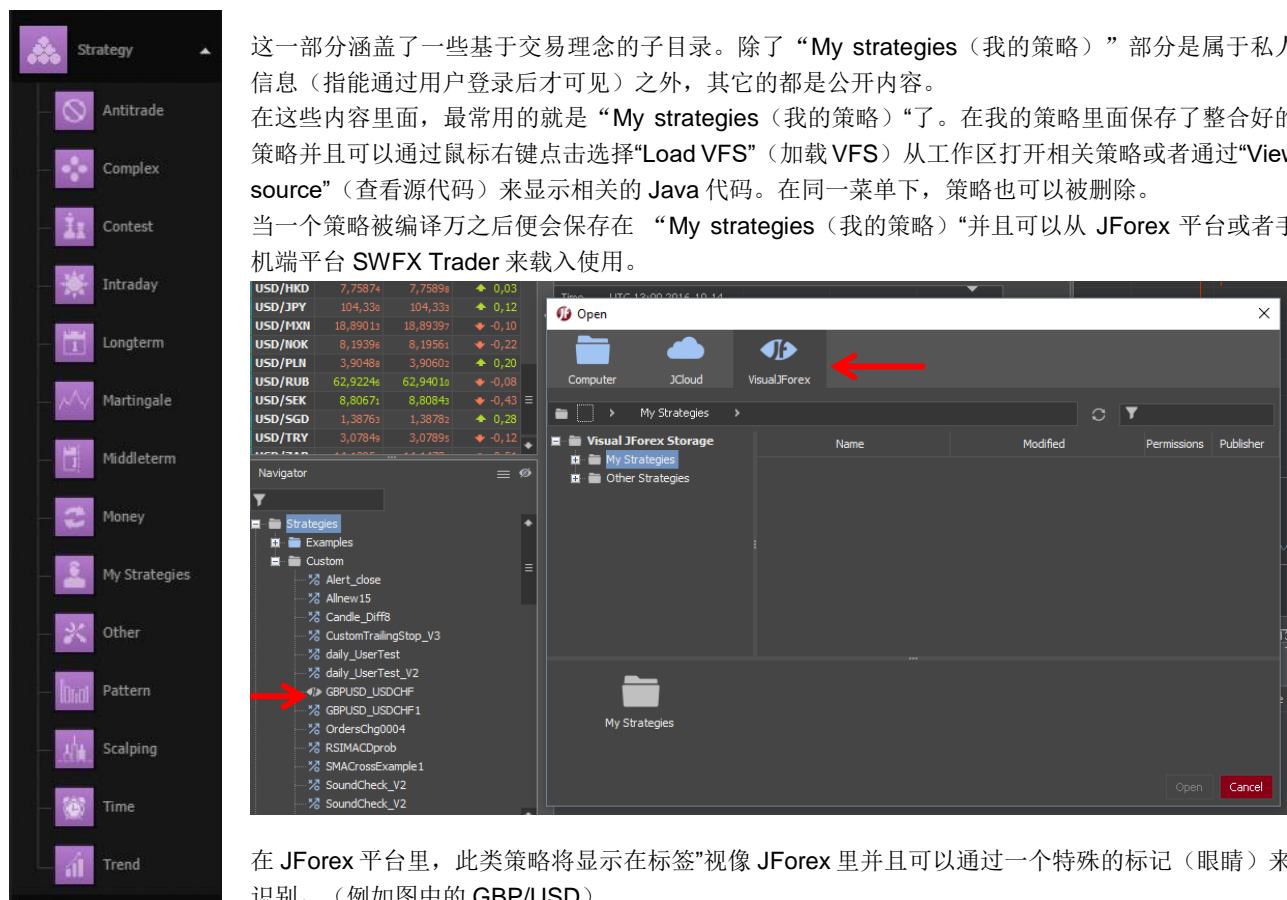
每一类指标下的技术指标以其简称显示。全名和具体的介绍在点击相关模块左上方的感叹号后可以查看到。如果某些技术指标无法显示此类信息, 那么也可以选择从 JForex 平台里查看。



几乎所有的技术指标都会显示两个参数的设定。第一个是开仓的方向, 客户可以选择指标是基于哪个方向计算的, 买方出价或者卖方出价。第二个参数是基于价格计算的, 其会被显示为 “Applied price” (被应用的价格)。一些方法提供简单的和整合的价格输入值。

5.2. 策略:

以紫色显示的策略部分显示了以主题归类的策略群组。

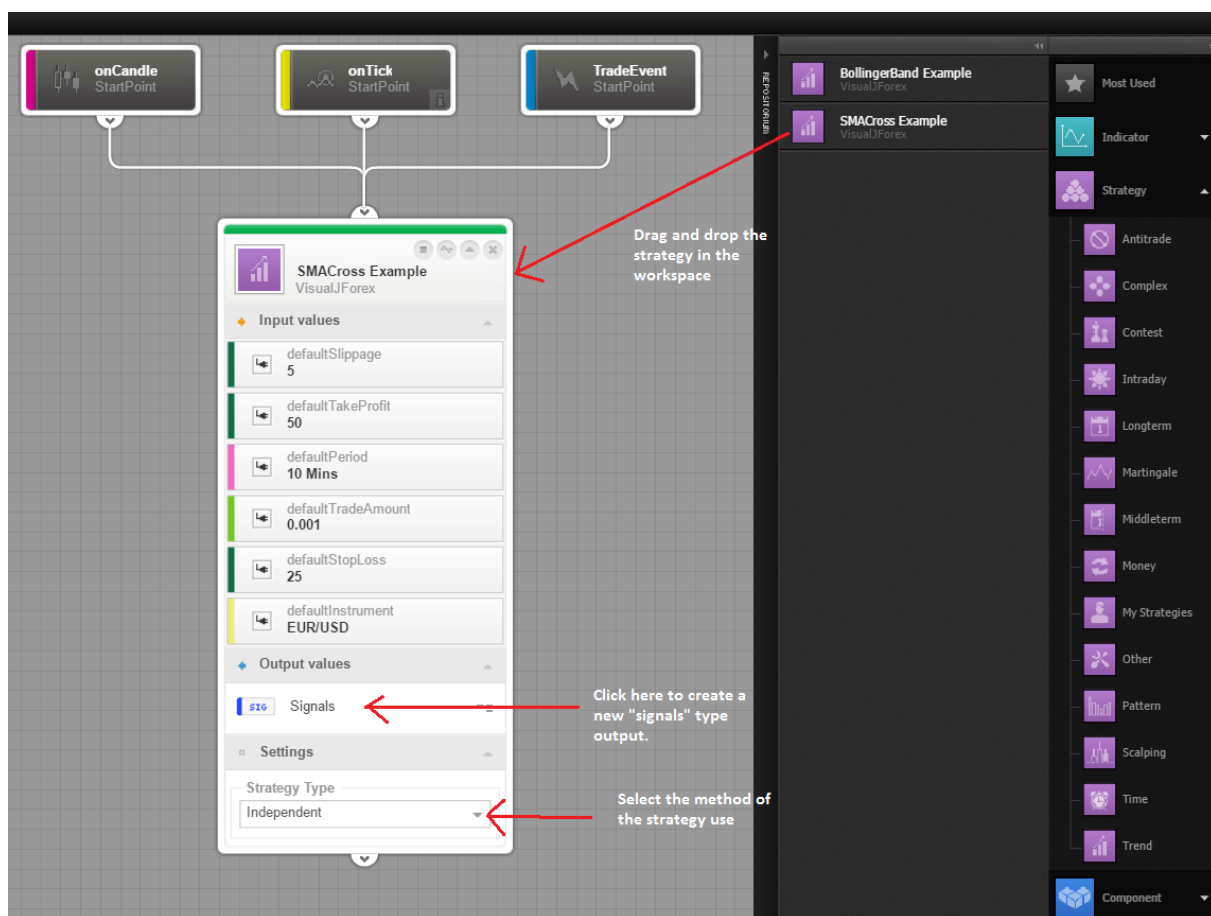


在 JForex 平台里，此类策略将显示在标签“视觉 JForex”里并且可以通过一个特殊的标记（眼睛）来识别。（例如图中的 GBP/USD）

什么是策略应用为模块？

这是一种简化的方法来优化工作区并将整个策略以单一模块的形式展现用于生产信号甚至直接运行。信号是除了仓位相关信息和订单创建之外的一种由策略发起的交易指令的捕获。当使用现成的策略从所生成的交易指令中获益以及添加额外的条件和优化时，此类方法也是很实用的。在一击成交中更改策略参数也变得更加容易因为它们策略模块中属于输入型变量。

使用策略模块并生产信号，只需简单的拖动策略到工作区，建立一个新的“Signals(信号)输出并在模块设定中选择”generate signals(生产信号)”参数即可。



为了能使其正常使用，策略模块是必须同时和三个起始点互相连接的。（基于蜡烛，基于 Tick 以及基于交易事件）

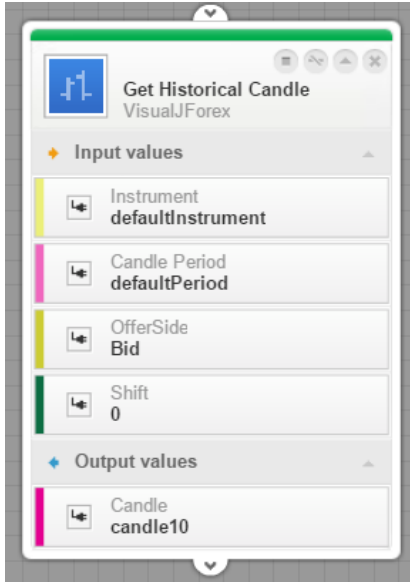
输出型变量为“Signal”类型(段落 6.2.2) 是一个用来保存和此类信号相关的信息的数据数组。此类变量是“嵌入的”并且在“Loop Viewer （循环查看器）”模块的帮助下获得数据分解。为此，连接一个循环查看器并使用新创建的输出值“Signals”作为第一输出变量。在循环查看器里创建一个新的输出值来分解和保存你的信号(循环流程详见段落 **Error! Reference source not found.**)。

当以此类方式运行策略时，用户可以选择“Independent (独立)”模式或者“Generate signals (生成信号)”模式；第一种模式类似于使用正常的（全部模块）来运行策略而第二种模式只生成没有有效交易指令的交易信号。信号是策略使用的信号指令的后端通知；这样用户就可以使用信号作为后续条件的触发点了。

5.3. 组件

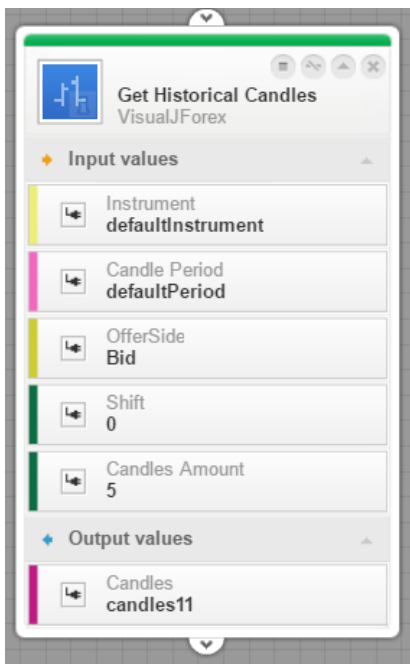
组件位于数据功能库最后的蓝色子目录里；它们也按照类别排序了。组件是用于构建逻辑型条件和语句以及数学和交易指令操作的功能技术型模块。

5.3.1. 信息:



Get Historical Candle (获取历史蜡烛数据)：获取当前正在使用的蜡烛或者之前已经结束的历史蜡烛信息。筛选可以通过参数“Shift”（位移）来实现。默认情况下为 0，代表当前的蜡烛。用户可以选择交易产品、蜡烛时间段以及买卖方向；输出值以一组自动生成的数列变量保存，其类型为“Candle”（蜡烛）。(6.2.2)

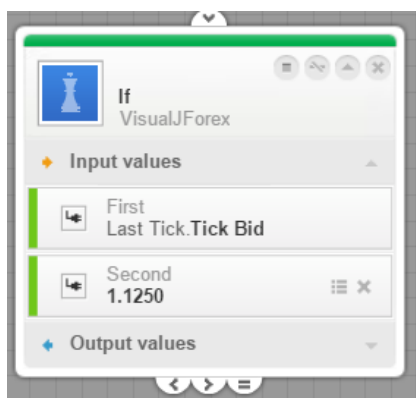
输出值将适用于“Auto created variables”（自动创建的变量）并且会包括开始、最高、最低、结束以及交易量、蜡烛时间段和蜡烛起始时间。



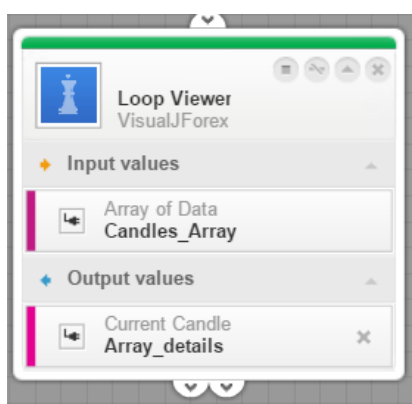
Get Historical Candles (获取多个历史蜡烛数据)：与之前的模块是处理单个蜡烛信息一样，该模块在“Candle Amount（蜡烛数量）”输入值的帮助下可以锁定多个蜡烛数据并进行使用和保存。输出值是一个数组，将相关信息存储在会用在“Loop Viewer”（循环查看器）里的内嵌式的变量中以此来获得新的变量分解。

此类模块会被应用在您需要使用大量蜡烛并且很难从许多单一的“Get Historical Candle（获取历史蜡烛数据）”模块中完成的情况下。

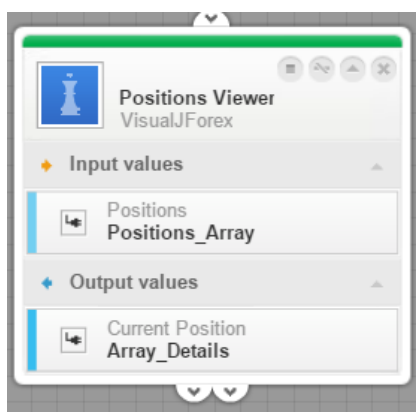
5.3.2. 逻辑型



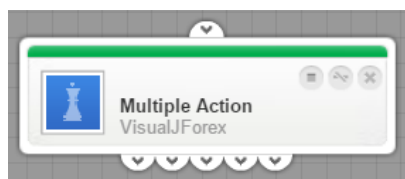
IF（如果）：其被广泛应用于编程来执行逻辑型条件；它同时处理两个输入值并支持许多类型的变量。该模块将在检测到变量类型不一致的情况下警告用户并建议作出相关更改。如果使用的是交易产品变量类型，那么模块将会自动更改输出口为两个（等于/不等于）并且第二个输出值会提供一个拥有所有可交易产品的下拉列表。此类变量类型如下：**Boolean, Instrument, String, Offer Side, State, and Command.** (详见段落 6.2)



Loop Viewer（循环观察器）：循环是编程中常用的基本编程思想。该模块执行循环或连续重复的指令序列，直到满足某个条件。它主要用于细分具有“**Candles（蜡烛）**”、“**Positions（仓位）**”或者“**Signals（信号）**”类型的数组变量。与模块“**Get historical Candles（获取历史蜡烛数据）**”相关联，它将获取从相关蜡烛中收集的具体数据。当连接策略模块时 (5.5.2)，输出值“**Signals（信号）**”将作为输入值并且其完整的细分数据将被保存为新的输出变量。当“**Position（仓位）**”变量被用作输入值时，其充当的角色作用将和“**Position Viewer（仓位观察器）**”一样。循环监测使用的是左侧的输出点而当使用右侧的输出点时将是循环（loop）的结束。

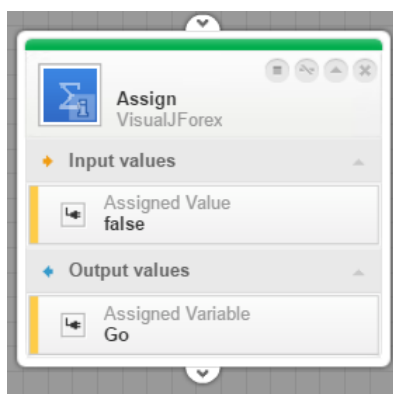


Position Viewer（仓位观察器）：它允许用户执行连续重复的指令序列，直到达到一个给定的条件。针对于仓位的数组数据，该模块将执行和循环观察器相同的功能。区别在于仓位观察器将针对于仓位变量。通常情况下，输入值取自左侧面板里的“**Position Info（仓位信息）**”部分。这将为用户提供一个处理既定仓位状态（已开、待定或者两者皆是）的选择。为了分解使用的输入值，用户需要创建一个输出变量；迭代的结果将被保存为新的标量并保存在“**User's variable（用户的变量）**”部分内。

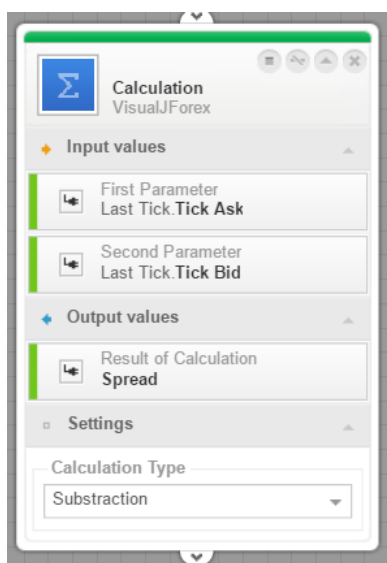


Multiple actions（多项操作）：这是一个简单的模块，帮助分流主线到多个支线（最多五个输出口）；它可以更进一步以倍数被重复分配。

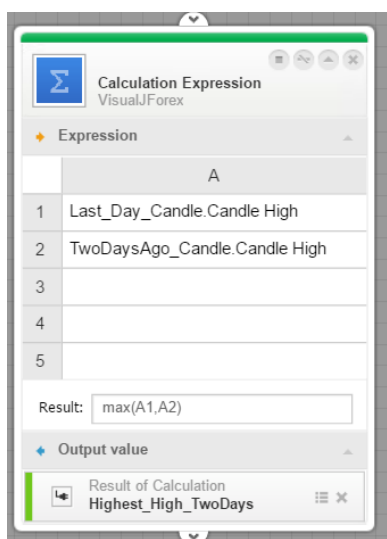
5.3.3. 运算



Assign（分配）：根据变量的类型给其分配一个既定的值。当第一输入值设定完之后，输出值会自动选择合适的变量类型。



Calculation（计算）：该模块用于基本的数学运算例如加减乘除法。计算的结果将用于后续的进一步使用。支持的变量类型有 Double, Integer, Date & Time (6.2.1)



Calculation Expression（计算表达式）：用于复杂的计算和数学函数，该模块以类似于 Excel 电子表格的形式展现。用户可以通过右击行/列表标题来添加行或列。支持的运算有：

"-" Subtraction

"+" Addition

"/" Division

"*" Multiplication

"Power(X, Y)" Returns the value of X to the power of Y

"%" Modulus

"Min(X, Y)" Returns the minimum between X and Y.

"Max(X, Y)" Returns the maximum between X and Y.

"Sqrt(X)" Returns the square root of X.

"Sin(X)" Returns the sine of X, where X is assumed to be in radians

"Cos(X)" Returns the cosine of X, where X is in radians

"Tan(X)" Returns the tangent of X, where X is assumed to be in radian

"Ln(X)" Returns the logarithm base e of X

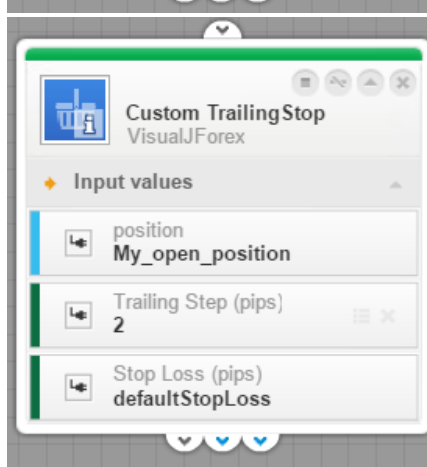
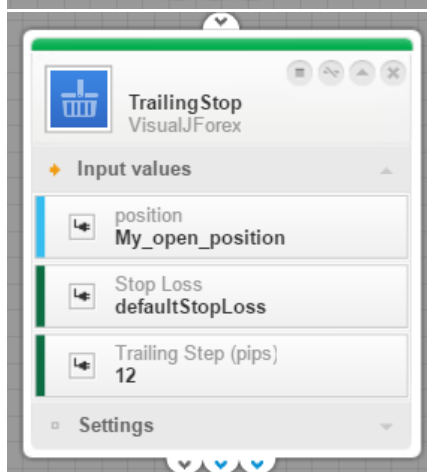
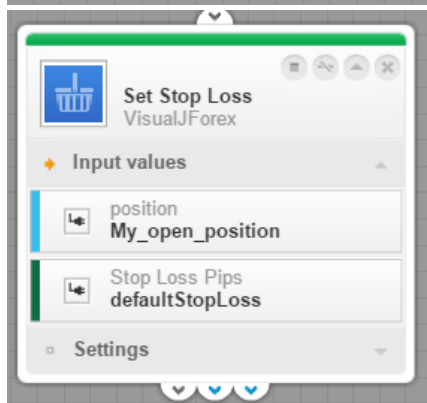
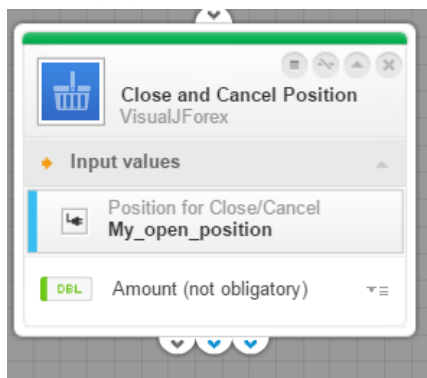
"Atan(X)" Returns the angle in radians between -pi/2 and pi/2 whose tangent is X.

"acos(X)" Returns the angle in radians between 0 and pi whose cosine is X.

"asin(X)" Returns the angle in radians between -pi/2 and pi/2 whose sine is X.

"exp(X)" Returns Euler's number raised to the power of X value.

5.3.4. 交易



Close and Cancel Position（关闭和取消仓位）：基于仓位的输入型变量，该模块可以关闭已开仓位或者取消挂单。如果需要部分平仓，那么第二栏“Amount（量）”里需要填写想要关闭的仓位量。

模块的输出点为：

- 无需基于订单的简单输出口
- 已平仓位或者已取消的订单 (正常状态)
- 订单取消被拒绝或者平仓被拒绝

Set Stop Loss（设定止损）：对于一个既定的仓位进行止损单的添加，该仓位的第一变量为“仓位”。

该止损单可以被设定为点数 (integer) 或者价格 (Double); 可以通过设定菜单进行选择。触发标准也可以被选择 (基于买方出价/卖方出价触发)。

模块的输出点为：

- 无需基于订单的简单输出口
- 订单递交已确认 (订单已建立)
- 订单递交被拒绝

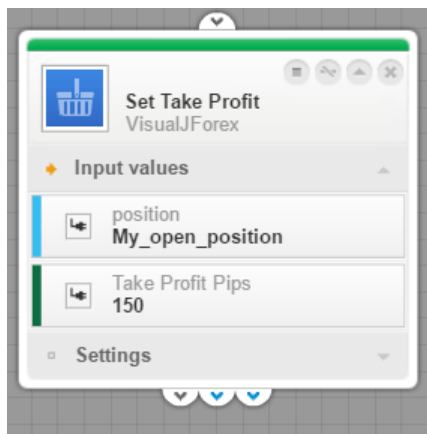
Trailing Stop（移动止损）：对于一个既定的仓位添加移动止损。初始的止损单被具体列在“Stop Loss（止损）”栏里；移动止损点数需要高于或等于 10 个点。和之前的止损模块一样，用户可以选择设定价格基于点数(integer) 或者基于已被计算的价格 (Double) 。

模块的输出点为：

- 无需基于订单的简单输出口
- 订单递交已确认 (订单已建立)
- 订单递交被拒绝

Custom Trailing Stop（自定义移动止损）：为了建立一个小于 10 点的特定移动止损，用户可以配合仓位以及其初始止损单来使用此模块。此模块输出点和之前的交易模块相同。

止损和移动止损模块需要建立在可以全面被策略管理的基础上；追踪过程是基于策略的而非服务器。一旦满足价格条件，策略应该达到可以执行这些模块的状态。

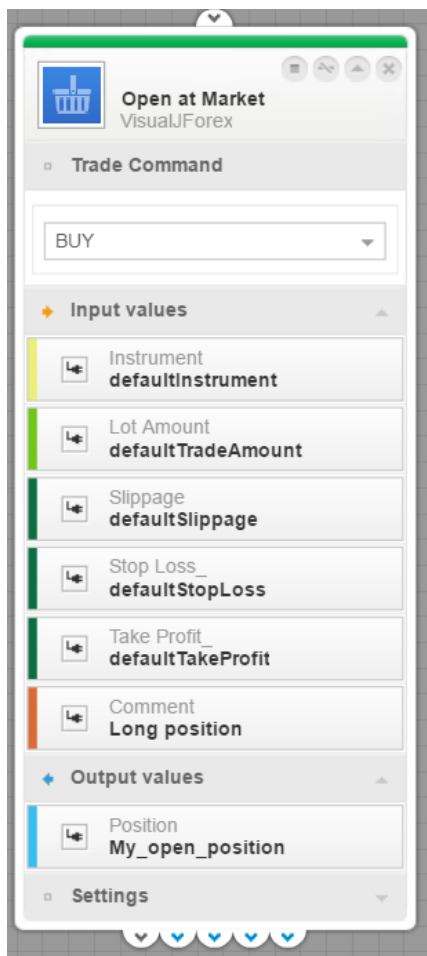


Set Take Profit（设定止盈）：对于一个既定的仓位进行止盈单的添加，该仓位的第一变量为“仓位”。

该止盈单可以被设定为点数 (integer) 或者价格 (Double); 可以通过设定菜单进行选择。触发标准也可以被选择 (基于买方出价/卖方出价触发)。

模块的输出点为:

- 无需基于订单的简单输出口
- 订单递交已确认 (订单已建立)
- 订单递交被拒绝



Open at Market（建立市价单）：该模块用于递交市价订单。第一次使用的时候，模块将显示默认的变量参数，当然，在输入型和输出型变量处也可以添加新的变量数据。

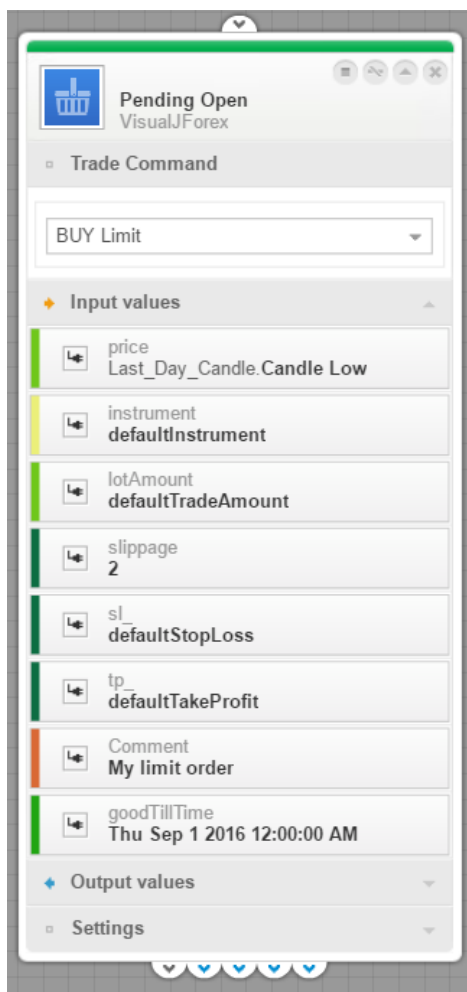
订单方向可以在 **trade command**（交易命令）的下拉菜单里进行选择。设置部分可以提供设定止损和止盈的价格是基于点数还是价格的选项。当止损/止盈不需要的时候，用户可以简单的通过点击变量右下角的“大叉”按钮来删除该变量; 这也适用于“**Slippage（滑点）**”变量。

默认情况下，交易量以百万计算（1 代表 1 百万，0.001 代表 1 千，等等）。

Comment（注释）部分并不是在模块刚刚放入工作区的时候就设定好的，因为它并不是一个必要输入参数；用户可以选择填写或者留白。“**comment（注释）**”将和仓位相关联，由此作为识别它的一种技术。

模块的输出点为:

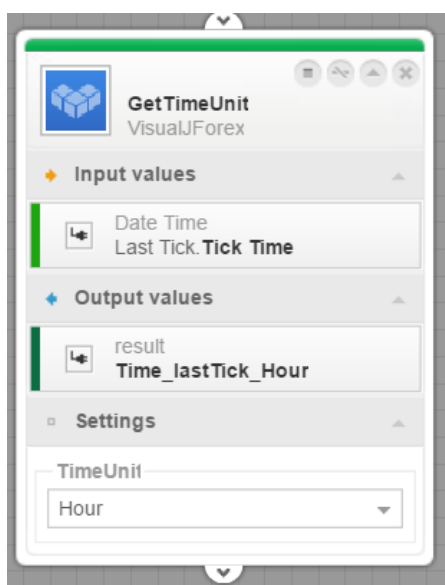
- 无需基于订单状态的简单输出口
- 订单递交已确认（订单已建立: 市价单的一个快速的状态，因为其将被立即发送到市场上）
- 订单递交被拒绝
- 订单被执行
- 订单执行已取消 (非常罕见的情况下执行会被取消)



Pending Open（待定建立）: 使用此模块通过交易命令的下拉菜单来建立入市订单: 限价单, 止损单（和触发标准选项一起）, 触价订单, 限价买入和限价卖出。除了一般的输入型变量之外, 该模块确实需要价格输入值但也有“有效时间”栏, 它和订单的有效性相关（列表中的最后一个输入型变量）。模块的输出点为:

- 无需基于订单状态的简单输出口
- 订单递交确认（订单已建立）
- 订单递交被拒绝
- 订单被执行: 当达到订单价格后以及订单被执行后。这是通过接收“FILL（执行）”信息来侦测的。
- 订单执行已取消 (非常罕见的情况下执行会被取消)

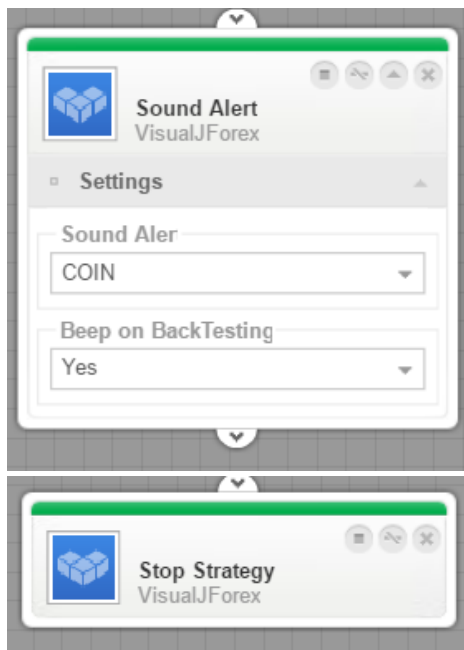
5.3.5. 其它



Get Time unit（获得时间数据）: 用来从蜡烛时间（蜡烛起始时间）或者从左边面板里可用的 Tick 时间来获取时间点。该模块通过一种方法来分解日期和时间, 使其成为分离的变量: 小时、分钟、秒, 一周中的某一天和一个月中的某一天。变量以 integer 形式保存。想要获取完整的时间格式, 可以使用多个模块。(Error! Reference source not found.)

一周中的天数被定义为 1 代表周日, 2 代表周一, 3 代表周二, 4 代表周三, 5 代表周四, 6 代表周五以及 7 代表周六。

➔ 日期和时间信息与蜡烛或者 tick 事件相关, 获取某个 tick 价格或者某个已开始的蜡烛时间。但是无法获取当前的 GMT 时间。



Sound Alert（声音报警）：为模块插入一个声音提醒并选择警报进行播放。默认设定下“Beep on back-testing”被设定为 No（不）， 如果想要听到警报声，可以将其转换到 Yes（是）。

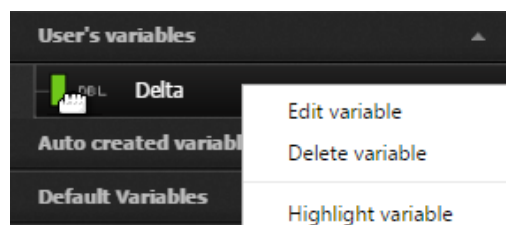
JForex 平台支持所有的警报类型。因此当策略代码导出至 Java 并在 JForex 里使用时，这些警报也是可以被正常播放的。

Stop Strategy（停止策略）：该模块用于永久性停止策略。

6. 变量

变量列表呈现在平台的左侧部分并以类型排列。

变量面板

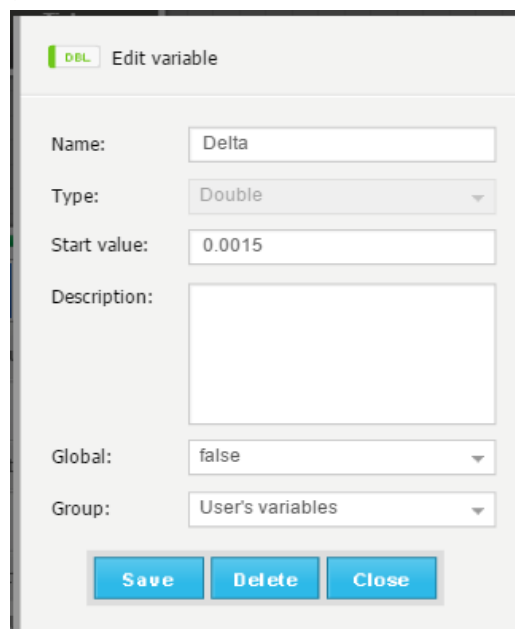


变量的菜单选项和设定

→**Edit variable (编辑变量)**: 编辑变量的名字、起始值、描述，通用设定或者群组（详见下文）。

→**Delete variable (删除变量)**: 永久性删除该变量

→**Highlight variable (着重标识变量)**: 这是一种着重标识已选变量闪烁的方法以此在工作区对其进行定位。一旦激活，变量的识别颜色将在左侧部分闪烁，它也会在合适的模块里吸引用户的注意。



变量的属性可以从上面的窗口里进行编辑。变量的类型无法被更改; 如果用户需要，可以创建一个新的变量。

“Global (通用)” 设定可以展示默认变量参数，因此它可以在启动策略之前快速的从弹出的窗口进行编辑。

6.1. 变量面板

变量部分在工作区左侧显示；其显示的变量可以用于填充模块。变量按照以下类别进行分类：

6.1.1. User's variables（用户的变量）：

这个部分适用于由用户创建的自定义变量。任何由用户创建的变量将被自动保存在该部分。一旦变量被创建，它可以从菜单选项里通过右击来编辑或者删除。

6.1.2. Auto created variables（自动生成的变量）：

有些模块会自动生成他们的变量输出值，该类型将被自动保存在此部分。比如，当使用一个指标的时候，输出值将自动生成并且经常以“Double”的变量形式出现，其名字是随机的。用户可以重命名该变量，在这之后，其将被保存在用户的变量部分。

6.1.3. Default variables（默认变量）：

默认的变量数据将适用于此部分：

- 默认的交易产品
- 默认的交易量
- 默认滑点
- 默认止损
- 默认止盈
- 默认周期

6.1.4. Account（账户）：

该部分适用于书序交易账户的变量：

- Account ID（账户号）：交易账户的标识
- Account Currency（账户币种）：交易账户的基础货币
- Equity（本金）：账户可用本金。当策略在真实环境下运行时，它将获取真实环境下的本金。当进行历史数据回测的时候，本金由用户自行设定。默认情况下，本金被设置为 50,000 USD。
- Leverage（杠杆）：账户的杠杆设定。
- Margin cut level（保证金削减政策）：这是账户达到保证金削减政策时候的杠杆使用率。通常设置为 1:200。
- Over the Weekend leverage（周末杠杆）：这是账户的周末杠杆。通常设置为 1:30 单也可以根据用户的需求进行更改。
- Use of leverage（杠杆使用率）：真实情况下的杠杆使用率。
- Global account（非对冲账户）：该变量标示账户的性质是“对冲”还是“非对冲”（Global）模式。默认情况下，账户被设置为对冲模式而不是“非对冲”。

6.1.5. Position's info:仓位信息：

仓位信息是一个动态处理生成的仓位状态的部分。当此类变量应用于特定的模块例如“Position Viewer（仓位观察器）”的时候，它可以在策略启动之前获取仓位的状态。这里的状态是指订单/仓位状态以及通过“Position amount（仓位量）”变量获得的仓位数量。

- **All positions**（所有仓位）：该变量经常和“**Position viewer**（仓位观察器）”模块一起使用并鉴定账户的所有仓位（无论仓位或者订单的状态是怎样的）。
- **Open positions**（已开仓位）：该变量经常和“**Position viewer**（仓位观察器）”模块一起使用并反馈已开仓位的具体信息。
- **Pending Positions**（待定仓位）：该变量经常和“**Position viewer**（仓位观察器）”模块一起使用并反馈挂单的相关信息：止损，限价，触价。

6.1.6.Trade Event（交易事件）：

交易事件负责处理平台所接收的交易信息。它可以手动保存自定义变量里过去的的数据；当该部分和先前的交易以及订单的相关信息一起使用时将变得非常实际。交易事件变量是和起始点“**Trade Event**（交易事件）”一起使用的。顾名思义，该起始点只有在接收到交易信息之后才会被激活。

交易事件变量按照多级别排列，结合“**Last Trade Event**（上一个交易事件）”做为第一个仓位接收大多数近期信息的第一级别。

→ 如果策略没有处在运行状态，那么先前的交易信息是否无法被获取或者使用的。

6.1.7.On Candle（基于蜡烛）：

属于当前蜡烛的变量被保存在该部分。这个会同时包含买方出价和卖方出价，除了蜡烛周期和相关交易量以及交易的产品和蜡烛开始、结束时间之外，买方和卖方出价也都将包括开始价格，最高价，最低价以及结束价格。

→ 和当前蜡烛相关的上一个蜡烛信息仍将是变动的。对于当前的蜡烛唯一可以知道的是**开始价格**。结束价格、最高价和最低价肯定是需要等蜡烛完成之后才能确定的。

6.1.8.On Tick（基于 Tick）：

和蜡烛信息一样，tick 变量为：买方出价和卖方出价，交易量，交易产品和 tick 时间。

→ Tick 时间不是 GMT（格林威治时间）时间而是价格报价是的时间（GMT）。

6.2. 变量类型:

下述变量是基于客户角度以及其在视像 JForex 里的用法而创建的。

6.2.1. 常用变量:

平台里使用以下五种标准的变量:



Double: 用于价格、交易量和指标输出值。货币对的价格以 Double 形式保存; 例如, 距离 5.2 个点必须以 0.00052 的形式保存为 Double 变量类型。



Integer: 诸如止损和止盈的点数表示方式可以被设定为 Integer。此类变量类型不支持小数点。如果用户需要设定止损 5.2 个点, 那么就需要将其转换为 Double 类型并且在止损栏里使用价格而不是在交易指令模块里使用点数。(模块设定 **Error! Reference source not found.**)



Boolean: 逻辑型变量只需要执行两个值 (True/False)。它被广泛应用于用户定义的逻辑触发器。例如: 仓位可以为做多或者做空, 因此变量“多仓”可以是 True (是) 或者 False (否)。



String: 支持文字输入。附属在仓位上的注释被定义为 String。另外, 可以是文字和数字组合形式的仓位 ID 也将适用于此类型。



Date and Time (日期和时间): 以欧洲/美国日期&时间格式显示的日期和时间变量被显示在客户界面层级上, 但以 Linux epoch 格式显示的则在后端层级上。Epoch 时间格式简单的将时间以毫秒表述: 1 个小时等于 $90 \times 60 \times 1000 = 5400000$

6.2.2. 自定义变量:

以下变量应用于 JForex API, 它是杜高斯贝 JForex 编程中专用的数据库。

→ **数组是一系列用于描述相关元件整合的数据。以下定义 Tick、蜡烛、仓位和信号变量为数组, 因此它们包括了相关数据变量的整合。**



Tick: 这是一组包括 Tick 的买方出价和卖方出价、相关交易量、交易产品、日期&时间的数组。



Candle (蜡烛): 蜡烛数组包括 OHLC 价格、蜡烛期限、交易产品和日期&时间。



Position (仓位): 仓位数组保存了既定仓位的相关信息包括仓位号、交易量、平仓价格和时间、建仓和成交时间、开仓价格以及更多其它内容。当“Position (仓位) 变量类型被创建后可以看到总的列表内容。此类变量大多数被用于”Position viewer (仓位观察器)”模块来获取相关的前一个/当前仓位信息。



Instrument (交易产品): 此类变量是关于交易产品的。当使用此类型创建一个新的变量时, 用户会被要求从下拉菜单里选择交易产品。



State (状态): 订单或者仓位的状态由该变量收集; 它被典型的用于 “Position (仓位)” 数组并动态的接收订单的当前信息。订单或者仓位状态的可用值为:

Created (已创建): 当订单刚刚被递交并处于前段水平

Opened (已打开): 当订单抵达了服务器并已被保存

Filled (已成交): 订单已成交因此变为已开仓位

Closed (已关闭): 当仓位以被关闭

Cancelled (已取消): 当某个订单已被服务器或者用户取消



Period (期限): 该变量属于蜡烛数组; 它使用了蜡烛的标准期限: 10 秒、1 分钟、5 分钟、10 分钟、15 分钟、30 分钟、1 小时、4 小时、1 天、1 周和 1 个月。不在上述列表中的其它的蜡烛期限将不适用; 自定义期限需要解决办法。



Command (命令): 该变量可以处理以下订单类型: 市价做多、市价做空、限额买入、限额卖出、止损买入、止损卖出, 基于买方出价的限额买入、基于卖方出价的限额买入、基于买方出价的止损买入、基于卖方出价的止损卖出、限价买入和限价卖出。



Message (信息): 它指的是由平台接收的借助变量 “Message Type (信息类型)” 分类的信息。



Message Type (信息类型): 是一个包含由平台接收的交易信息类的数组以及一些额外的还没有被完全实施的信息类型。此类变量类型可以在 Trade Event (交易事件) 部分找到。

- 已实施的信息类别为:

Position Rejected (仓位被拒绝): 当某个订单被拒绝的时候;

Position submitted (仓位已递交): 当服务器接收某个订单的时候;

Position Filled (仓位已成交): 当 “FILL (成交)” 被平台接收的时候;

Position Fill rejected (仓位成交被拒绝): 当某个执行被取消的时候;

Position Close rejected (仓位关闭被拒绝): 当某个平仓请求被服务器或者市场拒绝的时候;

Position closed (仓位已关闭): 当某个仓位被成功关闭的时候;

Position change (仓位变更): 当某个订单被成功编辑的时候;

Position change rejected (仓位变更被拒绝): 当某个订单编辑请求被服务器拒绝的时候

- 目前而言, 未被支持的信息类别为:

Position Merged (仓位已合并): 当两个 (或更多) 仓位被成功合并的时候;

Position Merge rejected (仓位合并被拒绝): 当合并请求被服务器拒绝的时候;

Mail (邮件): 处理在策略配置中的邮件请求; (比如当某个邮件请教发送的时候)

News (新闻): 由平台接收的市场新闻;

Calendar (日历): 已接受的经济日历事件;

Notification (通知): 已接受的平台通知;

Instrument status (产品状态): 产品状态 (如有变更)

Connection status (连接状态): 当前账户的连接状态 (如有变更)



Offer side (交易方向): 指的是指标计算方法是基于卖方出价还是买方出价。



Signal (信号): 信号是当整个策略作为单个米快链接的时候特别适用的变量数组。(更多关于 “策略模块” 可以在 “如何建立策略” 部分查看) 此类变量触发交易信号的时候无需有效的发送交易指令。它是为特定使用信号交易或者其它类似交易方式创建的。当建立复杂概念的时候它也简化了视像 JForex 的作用。

信号数组包含 “Signal Type (信号类型)” 以及与 “Signal Position (信号仓位)” 中表示的相关仓位的信息, 其中列出了与 “Position (仓位)” 相同的变量类型。



Signals（多个信号）：在多数情况下，该变量的创建是基于策略被用作磨料来保存其交易信号的。该数组被嵌入，并且在“Loop Viewer（循环查看器）”的帮助下获得详细的分解数据。然后将输出值列在“Signal（信号）”变量里（如上所述）。



Signal Type（信号类型）：信号数组的一部分，信号类型指的是有既定策略创建的交易命令。信号类型为：仓位买入、卖出、合并、关闭和订单变更。



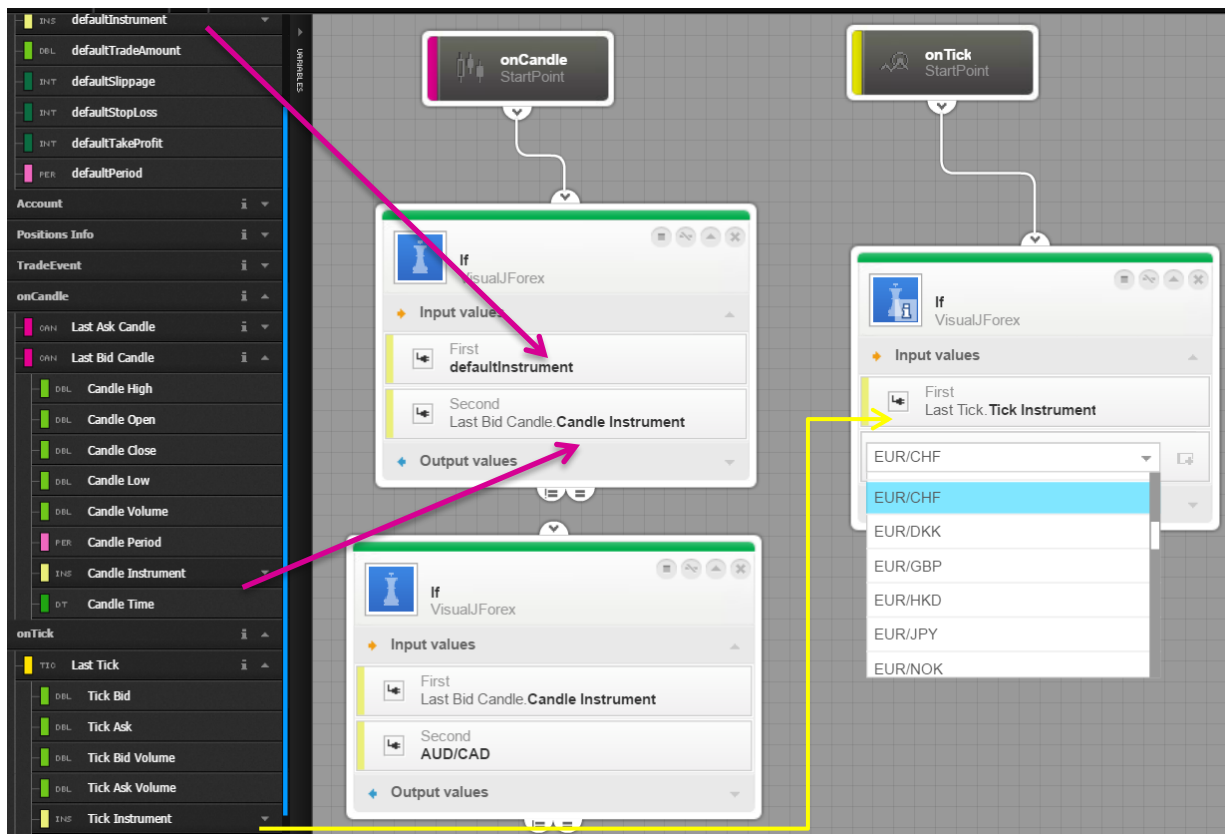
Moving average type（移动平均类型）：这是关于 MA 类型的特定变量；它采用在下拉菜单（列举了 12 中移动平均类型）里列出的特定值。



Applied price（应用的价格）：和之前的变量一样，应用的价格也是配合某个既定指标使用的并且是关于指标计算公式里使用的价格类型（平仓价、开仓价、中间价、加权平仓价等等）。该变量帮助动态的改变指标计算方式。

7. 如何

7.1. 如何做一个策略的指定



7.1.1. 指定交易产品:

默认情况下，视像 JForex 的蜡烛方案是使用 EURUSD 的，但我们是推荐指定其它交易产品的，尤其是当策略准备用于 JForex 平台的时候。如果策略没有指定某个产品，那么当在 JForex 里使用的时候，它将会随意交易某个交易产品。其背后的技术原因是关于“On Candle（基于蜡烛）”和“On Tick（基于 Tick）”方法是用在 API 里的并且默认指定所有的交易产品。

指定某个交易产品涉及过滤掉其它不需要使用的产品并只保留需要的那个。在上方截图里，第一个左上方的模块以“default instrument（默认交易产品）”变量的形式以及“last Bid Candle instrument（上一个买方出价蜡烛的交易工具）”（也可以是卖方出价蜡烛）来筛选交易产品。然后筛选通过连接 IF 条件以及其“Equal（相等）”出口。下方的第二个 IF 条件也是过滤掉不需要使用的交易产品的方法但是无需“Default Instrument（默认交易产品）”变量，蜡烛交易产品会成为第一变量而当点击第二输入值方框时会显示下拉菜单。此方法是特别用于当策略实施于超过一个交易产品的时候。

当策略应用于 Tick 方案时，将使用相同的方法但是相对于蜡烛交易产品变量，用户需要使用 Tick 交易产品来代替，让其显示在右上方的模块里。

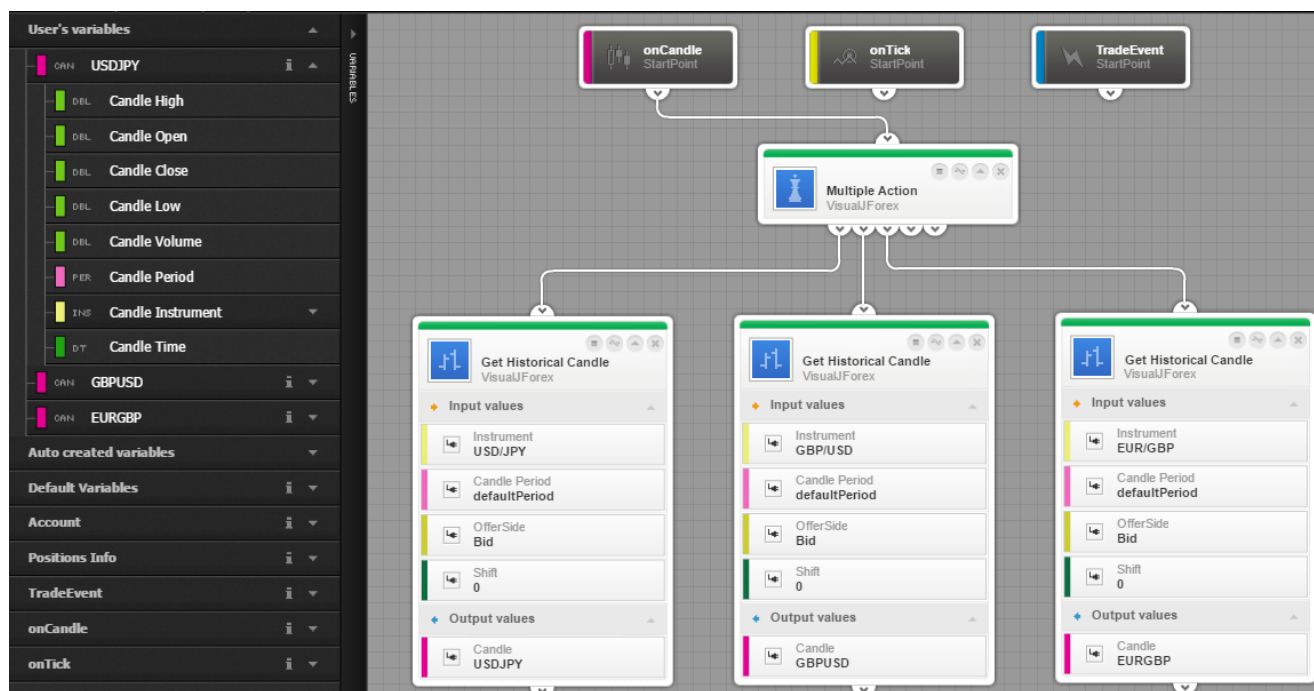
7.1.2. 指定期限:

指定期限和上面所描述的指定交易产品类似，不同之处在于使用了“Last Candle period（上一个蜡烛期限）”和“default period variables（默认期限变量）”来过滤掉不需要的蜡烛期限。原因在于“On Candle（基于蜡烛）”的方法会自动获取所有交易期限，因此，其将被每 10 秒激活一次（因为这是默认状态下可用的最短期限）。自定义期限需要解决方案。

7.2. 如何使用多个交易产品

建立一个有关多个交易产品的策略是可行的，但是对其进行测试将比较困难因为它需要导出策略到 JForex 平台以此来同时运行在多个交易产品上。从技术上讲，建立此类策略是基于“Get historical candle（获取历史蜡烛）”的帮助来给每个交易产品使用之前的蜡烛信息。

根据交易产品的需要使用尽可能多的“Get historical candle（获取历史蜡烛）”并删除默认变量然后从下拉菜单里选择所需的交易产品。相对于使用自动生成的输出值，在下面这个例子中使用专门的数据；这将简化每个交易产品的变量认证。



在上述那些模块之前添加一个期限过滤器是可行的。

在左侧截图里，当策略启动后，变量被放入工作区来调试它们的参数。如图所示，OHLC 价格是基于每个交易产品显示的并且可以用于之后进一步的条件。

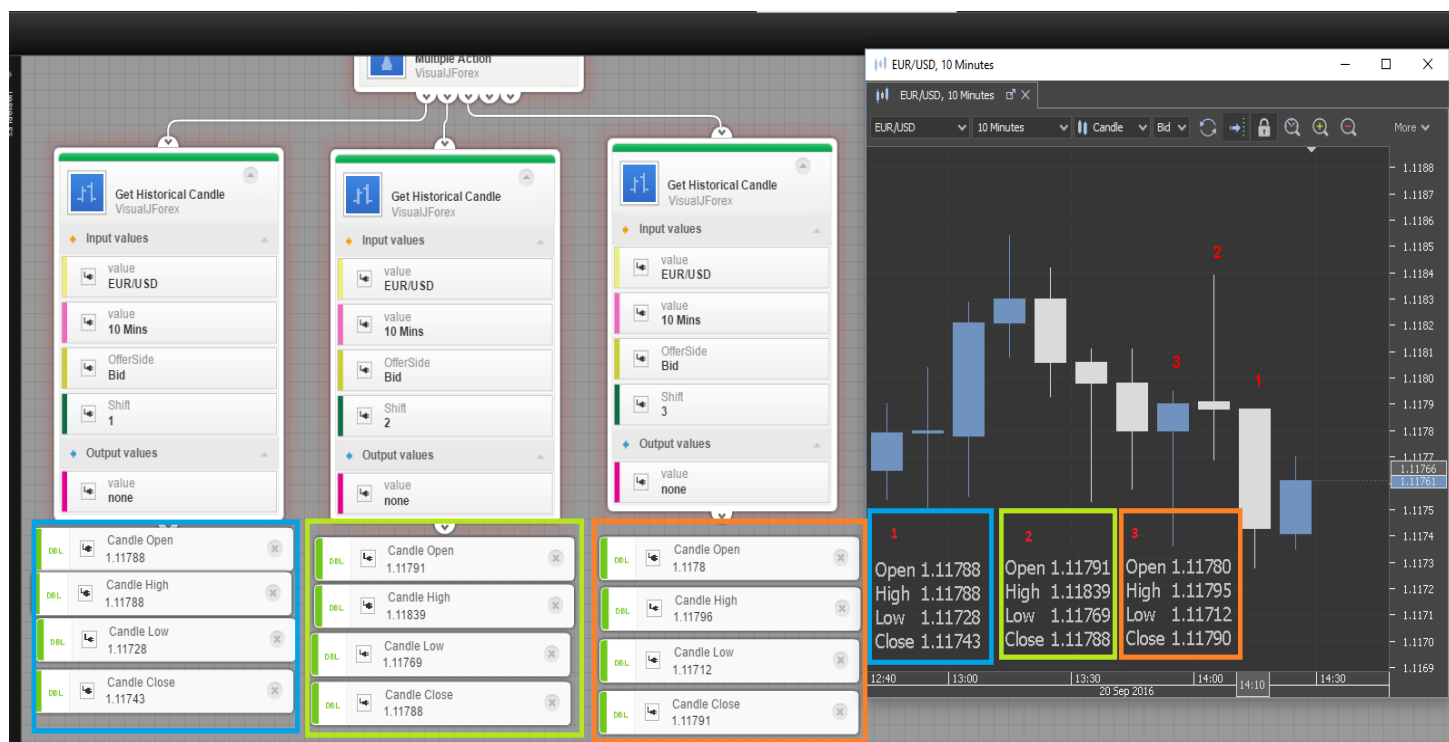
之前获取蜡烛数据的方法被广泛用于当策略理念应用于交叉交易产品逻辑的时候；换言之，比如当条件 A 被 EURUSD 验证后交易 USDJPY。因此，要交易一个既定的交易产，必须在另一个交易产品上验证交易条件。对于那些在几个交易产品上实施同样逻辑的策略，它会更简单的创建一个额外的策略文件并且在视像 JForex 层面更改交易产品：亦或者在 JForex 里运行或测试策略并且选择所需要的交易产品。

测试对个交易产品的策略无法在视像 JForex 里完成因为视像 JForex 实施一个单一的默认交易产品。此类策略只能在 JForex 里测试。

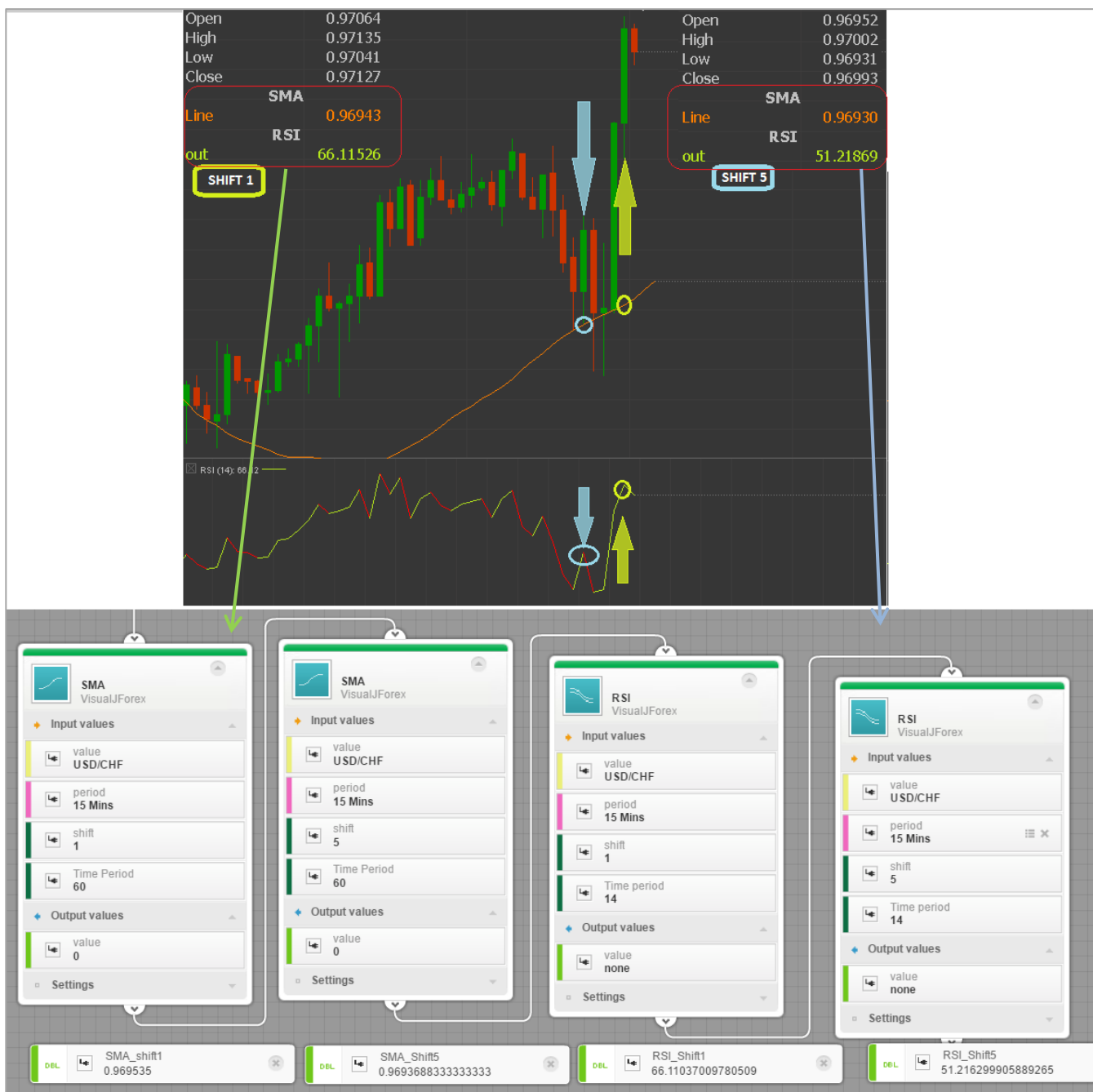
7.3. 如何处理移动值

Shift（移动）参数适用于几乎所有的指标以及“**Get Historical Candle(s)**”（获取历史蜡烛）模块。该参数在获取之前的某个或某些蜡烛的具体信息的时候显得尤为重要。

Shift 0 经常代表当前正在进行的蜡烛，因此唯一已知的信息只有开始价格，其余的信息将只会在蜡烛形成以及过期之后，当此蜡烛变为之前的蜡烛并伴随着新的蜡烛开始的时候才能获知。**Shift 1** 被用于获取之前一个蜡烛的相关信息，**Shift 2** 则是之前第二个蜡烛的信息，以此类推。



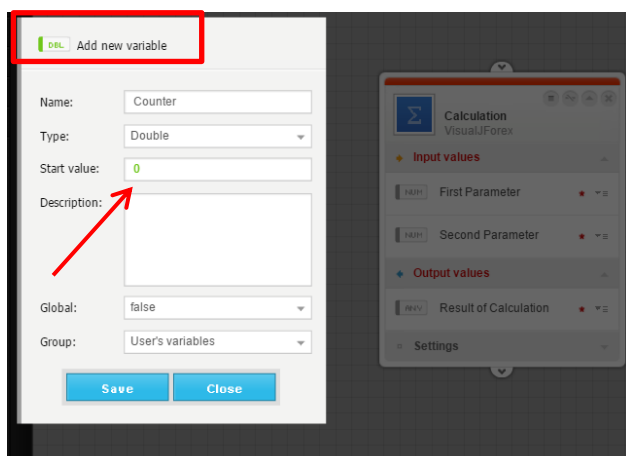
特别是当用户需要简则线型指标趋势的时候，**Shift**（移动）值将经常用于指标里；在比较先前的指标值和最近的一个指标值的时候，此类模式将被识别。该比较可以用来判断趋势是呈上涨还是下跌亦或者呈平稳趋势。



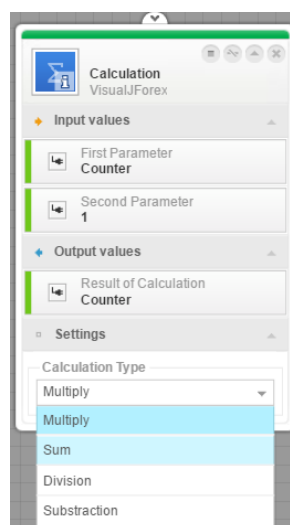
在上述截图里，SMA 和 RSI 指标使用的是 shift 1 和 shift 5，这将需要总计 4 个模块；当策略在真实环境下运行时，输出值变量获取他们的真实数据并匹配 JForex 指标的输出值。有些时候，用户可能会察觉到视像 JForex 和 JForex 输出值会有细小的偏差，这是由于应用在 JForex 指标输出值的四舍五入造成的。

7.4. 如何使用计数器

Counter（计数器）是一个可以帮助执行指令序列的简单工具。计数器的基本逻辑是在情况 A 发生的时候增加变量知道情况 B 出现，然后将计数器重新初始化为 0。在大多数情况下可以使用计算模块实现如下情况：



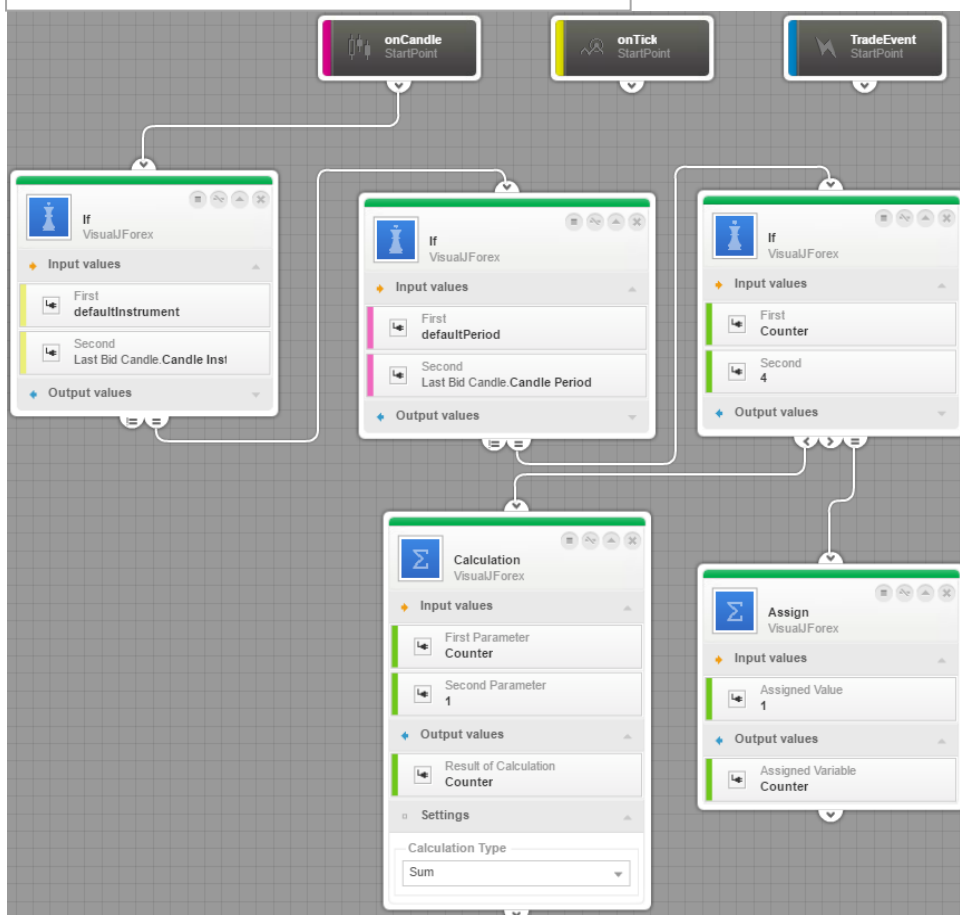
创建一个新的变量 (Double 或者 integer 类型) 并将其设置为 0。在该例子中，新的变量名字为“计数器”。



将新的变量 “Counter”（计数器）放入计算模块，创建另一个变量并设置为 1。
在结果栏（输出）里使用相同的计算器并选择 “Sum”（总和）做为运行类型。

→ “Counter”（计数器）已准备好使用了。

如何连接此模块将决定计数器的逻辑。



例子 1:

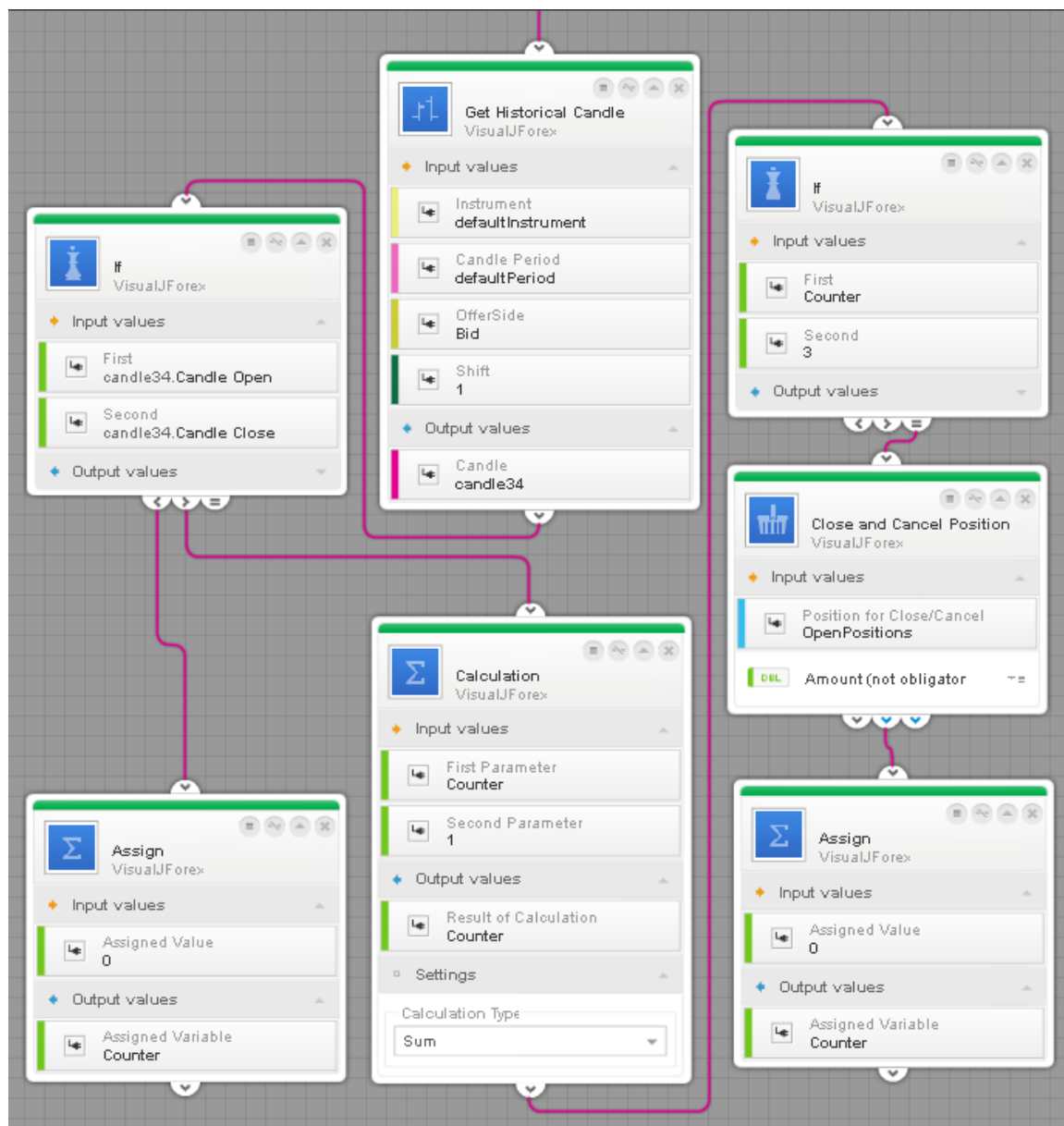
目的：设想您需要创建一个期限为 40 秒的蜡烛：

- 1/ 确定交易期限等于 10 秒。
- 2/ 按照上述内容创建计数器。
- 3/ 40 秒等于 4 乘以 10 秒，因此计数器需要设置增量直到 4 在这之后重置到 1。IF 条件非常重要，因为它将检查计数器的值是否达到 4。一旦达到之后，Assign（分配）模块会将其设置为 1，由此开启循环。

例子 2:

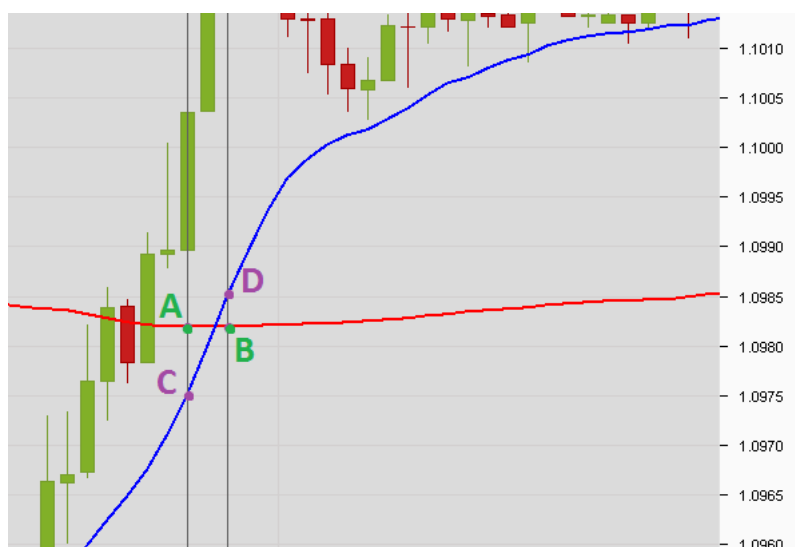
自从仓位创建后，在达到 3 个连续的红色蜡烛后关闭一个多头仓位。

- 1- 检查仓位的状态，在此情况下应该为 "Filled"（已建立）状态；
- 2- 获取之前的蜡烛信息并检查它是红色还是绿色；
- 3- 加入计数器来确保是否有连续三个红色蜡烛。



上方的模块负责处理策略部分，用于判断 3 个连续的红色蜡烛。“Get Historical Candle(s)”（获取历史蜡烛）模块加载在 IF 条件之后，用于检查仓位是否在已建立状态 “Filled”（已成交）状态。一旦之前的蜡烛被获取并判断为红色（Open>Close 开始>结束）且其增量值达到 3，那么在仓位成交后将传递到计数部分。计数器的计算频率是基于加载在策略起始部分的蜡烛期限过滤器的。计数器只有在先前的蜡烛为红色的时候产生增量，如果是绿色，那么计数器将重置为 0 因为没有达到条件。当该值达到 3 个之后，便确认了之前的 3 个蜡烛为下跌型（红色），因此仓位将被关闭然后计数器重置为 0。

7.5. 如何在两个指标之间建立交叉

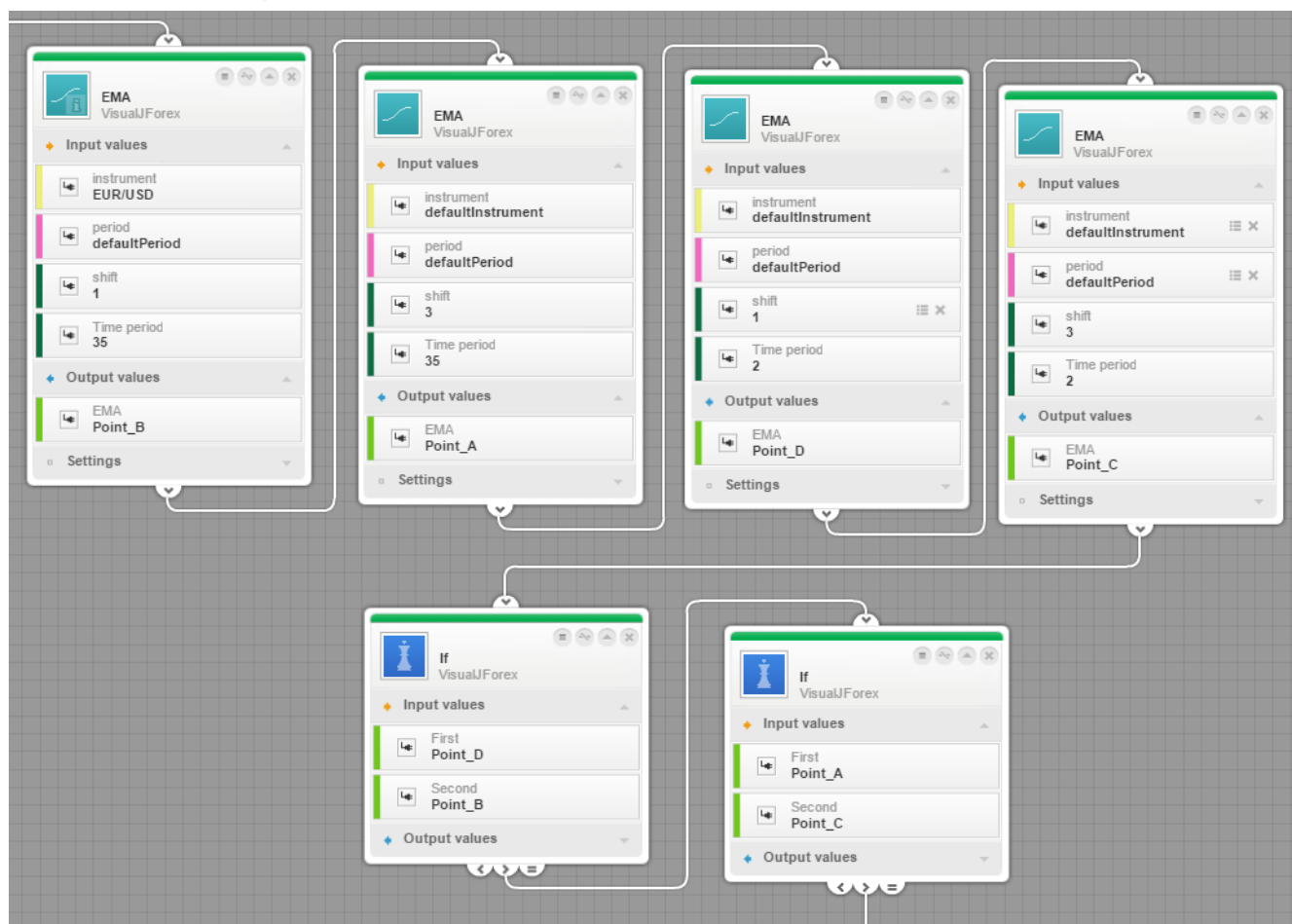


两条线之间的交叉点是基于四个点来定义的。在该例子中为A,B,C,D。该想法是使用这些点并核实它们的值来判断是否有交叉的发生。这将基于在指标模块里的“Shift”值的帮助。

交叉被定义为： $A > C$ 和 $D > B$ → 此定义适用于快速指标（蓝色）是否在上漲型趋势环境下与慢速指标（红色）相交叉。为了判断每根线条的趋势，可以添加一个额外的判断点： $A < B$ 和 $C < D$ 用于上升交叉，或者 $A > B$ 和 $C < D$ 用于下跌交叉等等。

第一步是添加合适的 shift 值到指标，需要重点提到的是，该模式的判断是基于 shift 值的选择：Shift 1 Vs Shift 2 相对于 Shift 1 Vs Shift 3 更为敏感：当数值之间的距离更长的时候会有更大的机会发生交叉事件。

对比基于如下 IF 模块：

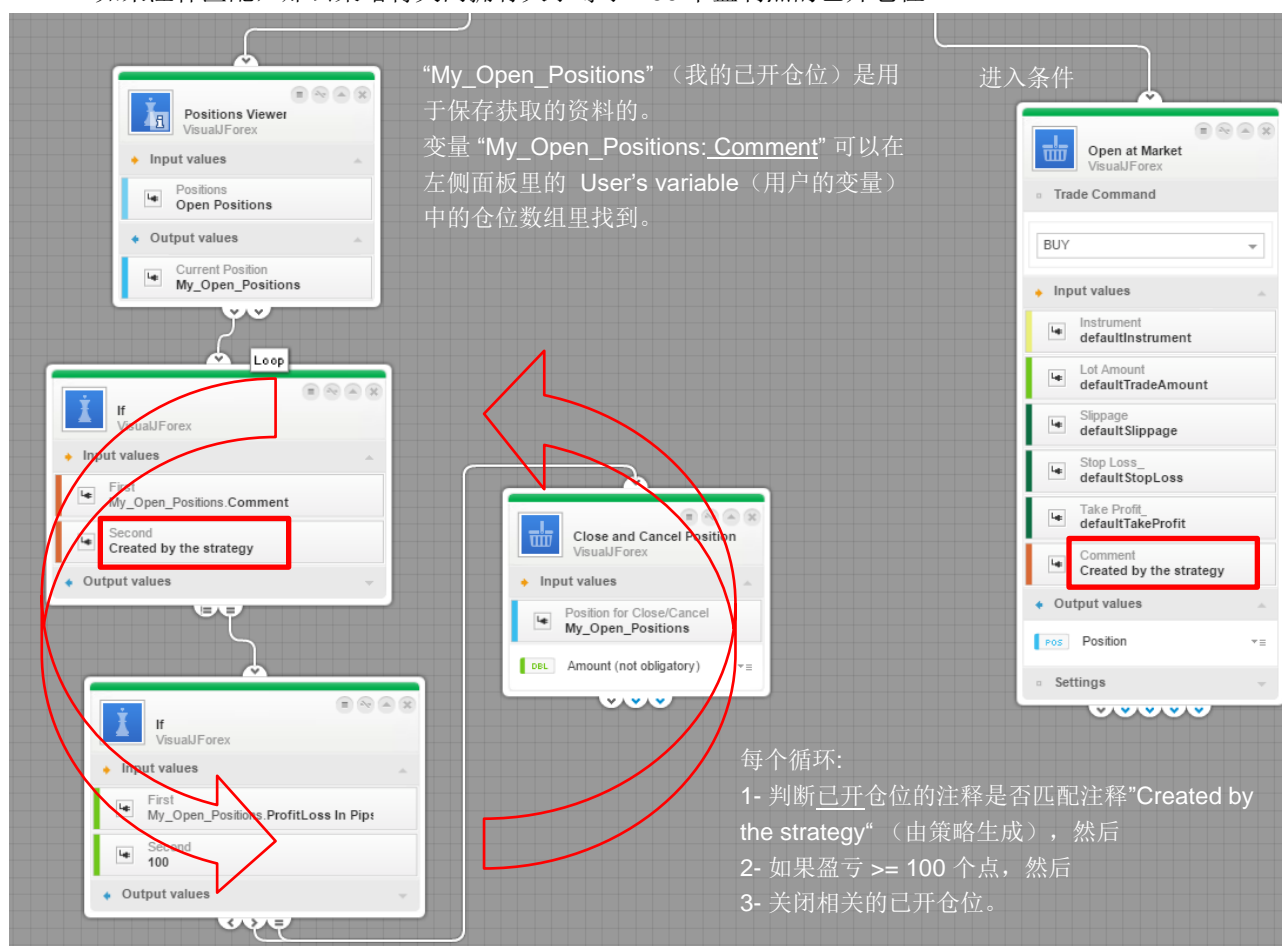


7.6. 如何判定和使用仓位

仓位由策略创建或者可以手动由“Position Viewer”（仓位观察器）鉴定；该模块充当任何使用循环功能的已成交订单的侦测器。一旦存在循环，便会容易获得已开仓位的方向、挂单、当前开始盈亏等等。

下面的例子目的是管理已开仓位并关闭那些由策略生成的已达到 100 个利润点的仓位。

仓位观察器模块可以直接连接到某个起始点。如果在中间有条件，用户需要确保这些条件不会影响仓位观察器模块的功能。循环频率是基于交易期限的。仓位观察器基于“comment”（注释）变量检查如果有任何由策略生成的已开仓位，“comment”变量必须在下方的每个交易命令里使用（Comment 由策略生成）。如果注释匹配，那么策略将关闭拥有大于等于 100 个盈利点的已开仓位。



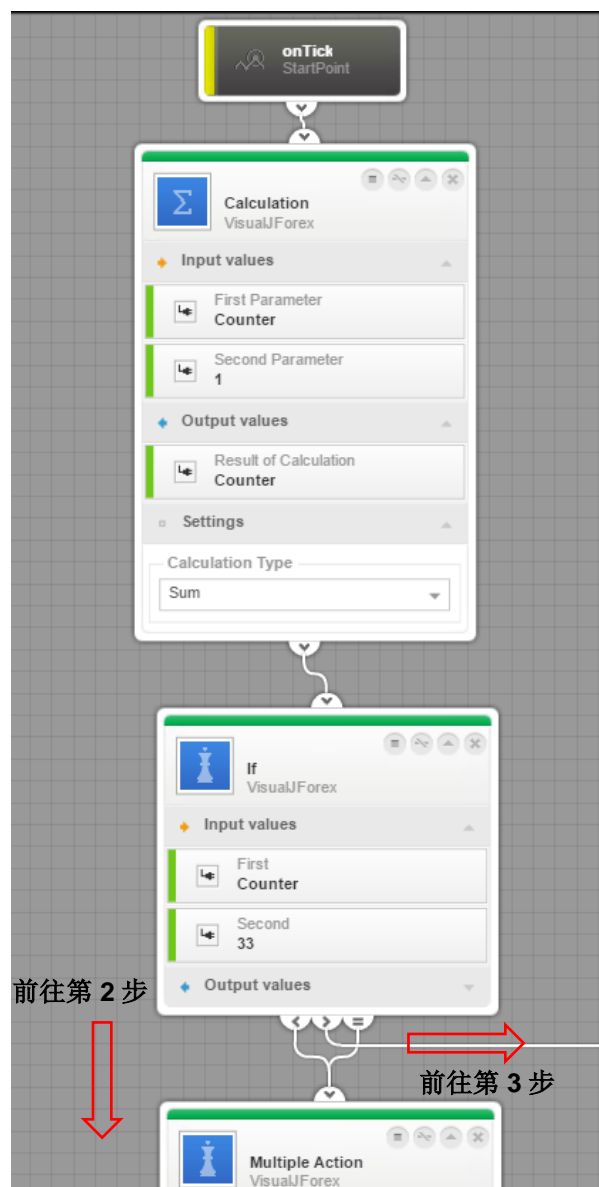
此外，用户可以配合另一个技术使用仓位：创建一个与每个模块“Open at market”（市价开仓）相关联的专用的仓位做为输出；验证是基于每个单独的仓位。该方法是用于基本策略的，当然，配合仓位观察器模块使用将更为有效。

“Position Viewer”（仓位观察器）的逻辑方案以及循环表现和管理与“Loop Viewer”（循环观察器）一模一样。唯一的区别在于循环观察器作为输入，支持更多的数组类型。

7.7. 如何建立 Tick 条形图（通过计数器）

Tick 条形图是一种基于预定 Ticks 数量创建蜡烛的图表类型。这个在 JForex 平台层面，通过偏好设置菜单是很容易实现的。然而通过编程在视像 JForex 层面来实现便需要使用另一种方案了。由于唯一的内置方法是蜡烛（10 秒至 1 个月），ticks 蜡烛是在计数器的帮助下建立的。在收到的每个价格后产生增量然后在达到目标 tick 数量之后停止。之后我们便需要添加另一套条件来获取和使用 OHLC 价格。

方案显示如下：



第 2 步和第 3 步显示在下一页

1/ 实施并设置计数器：

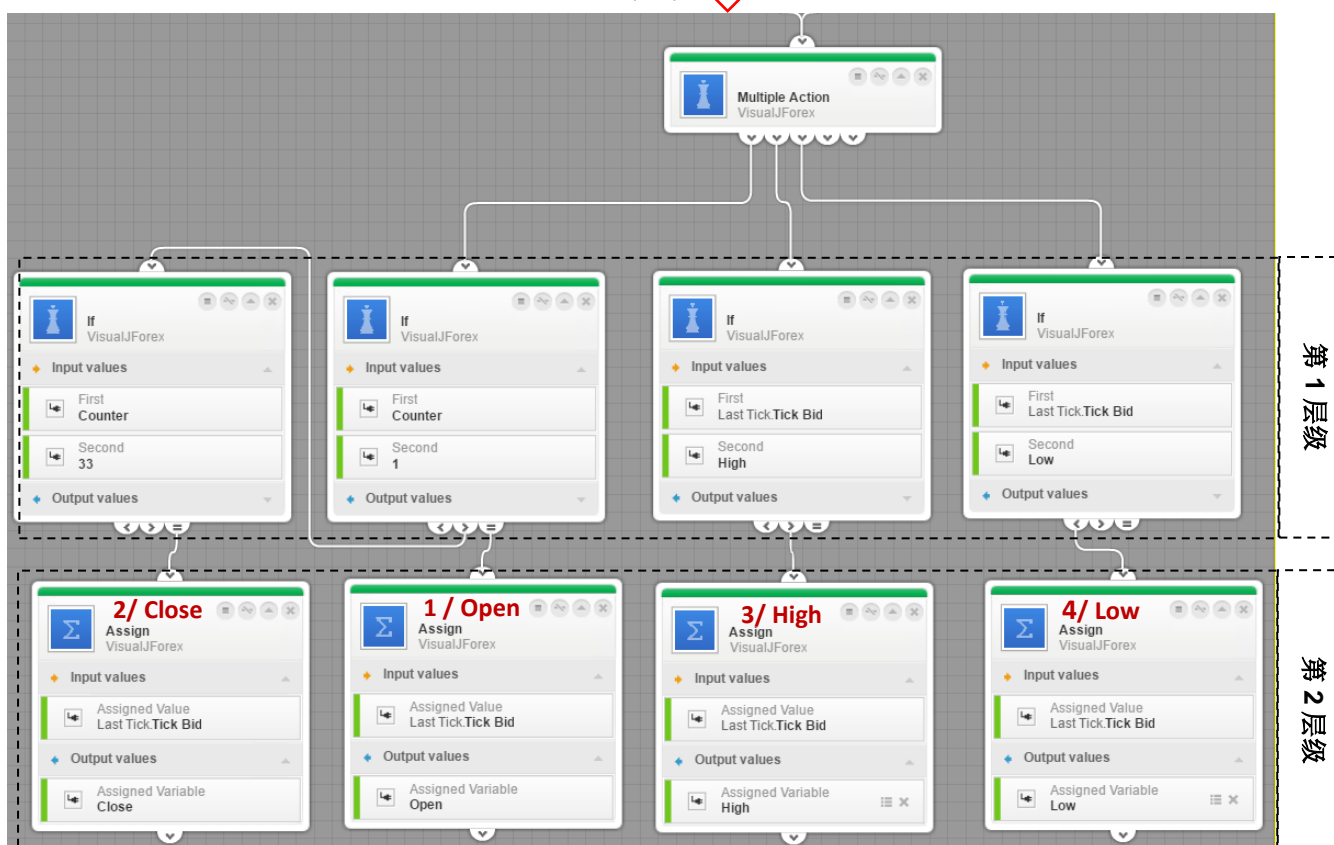
在该例子中，tick 条形图是基于 33 ticks 的。

计算模块是用于设置计数器的；需要注意的是同样的变量“计数器”是用于第一输入值和结果的。

IF 条件非常重要，因为基于计数器的值，它起到了分配流程的作用。当计数器值增加到 33 的时候，策略会继续第 2 步，当高于 33 的时候将执行第 3 步。

2/ 获取 OHLC

第2步 ↓



第2步 负责确认 OHLC 的价格。需要使用四个变量来依次保存数据。

- 第1层级: 新的变量开始、最高、最低和结束还没有被分配价格，它们配合 IF 模块使用来决定获取哪个数值：
- 第2层级: 一旦达到合适的条件，每个 OHLC 变量将获取以下价格：
 - 如果计数器等于 1 → 开始价格将被确定
 - 如果计数器等于 33 → 结束价格将被确定
 - 如果当前 tick 高于“最高”值 → 分配该 tick 价格为“High”（最高）
 - 如果当前价格小于“最低” → 分配该 tick 价格为“Low”（最低）
 - High 和 Low 变量将随着接收到每个新的 tick 更新，直到序列达到 33 ticks。

在此例子中，一旦第一序列的 33 ticks 完成，策略将覆盖先前的 OHLC 但是用户可以通过创建新的变量来保存之前的 OHLC。

3/ 结束循环并重置 OHLC 变量

第3步



4/ 在策略里使用 OHLC



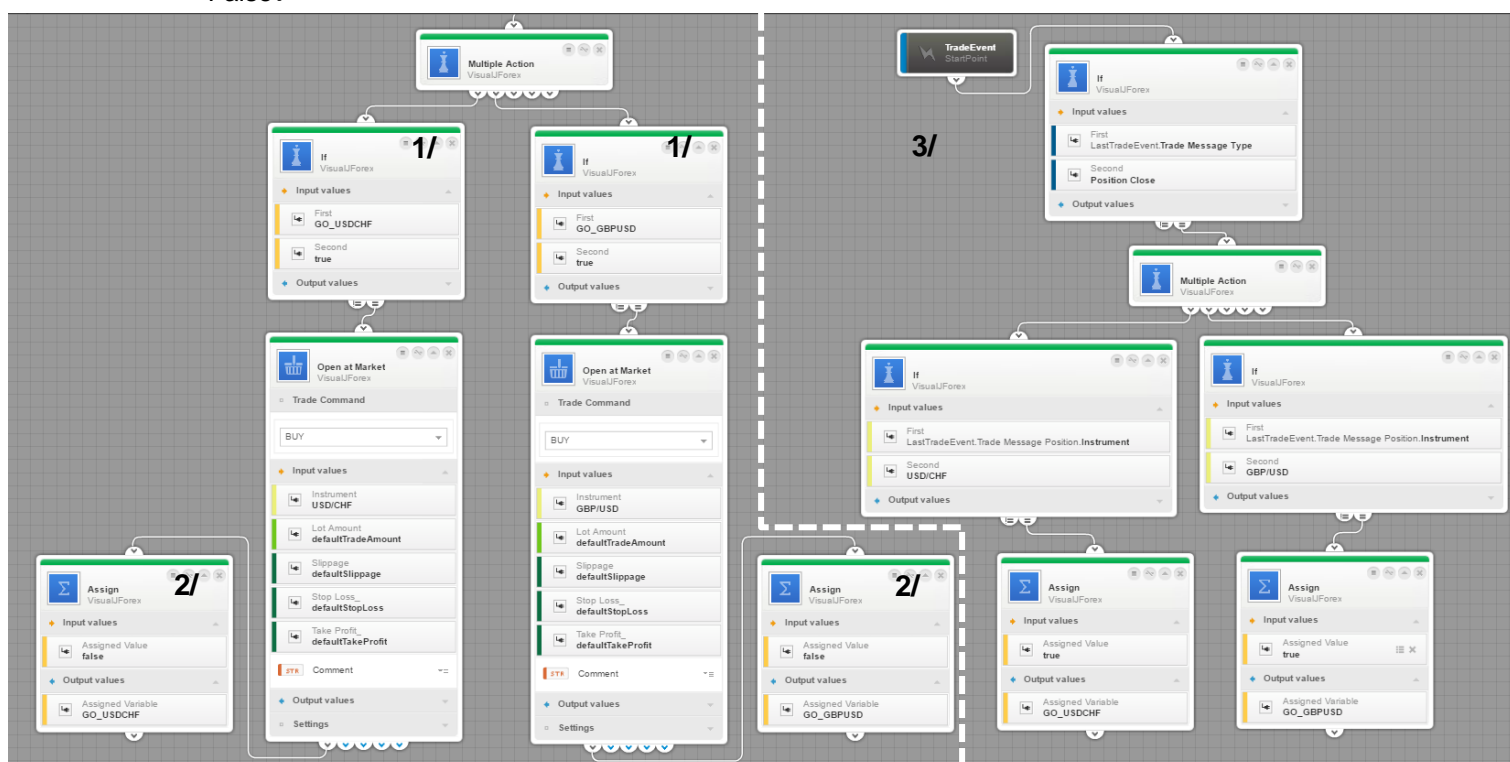
7.8. 如何处理逻辑型触发器

逻辑型触发器是由用户创建的，当条件 A 发生的时候获取一个既定值以及当条件 B 发生的时候转换到另一个数值。让我们来假设以下情形：

一个策略交易 2 个交易产品并且用户需要管理基于每个交易产品的已开仓位数量：要达成这个目的的一种方法为创建 2 个可以触发/阻止建仓指令的变量。

- 1/ 此类变量的起始值需要使该策略能够建立一个仓位；
- 2/ 一旦仓位被建立，变量会马上更改其数值来阻止下一阶段的仓位建立；
- 3/ 一旦仓位被关闭，同样的变量需要重置回最初的数值。

该流程图基于 2 个条件实行了数值的变更，因此我们可以选择一个 Boolean 变量来获取两种值，True 或者 False。



在左侧: 变量 “GO_USDCHF” 和 “GO_GBPUSD” 在开始的时候拥有“True” 值; 在最初开始的流程里，第一个 IF 条件将被通过然后每种交易产品在“GO_USDCHF” 和 “GO_GBPUSD”被切换为“False”的时候会有一个仓位被建立。当策略执行下一个流程的时候，IF 条件(1/) 将阻止进一步仓位的建立。其结果为: 每个货币对的已开仓位的最大数量不会超过 1。

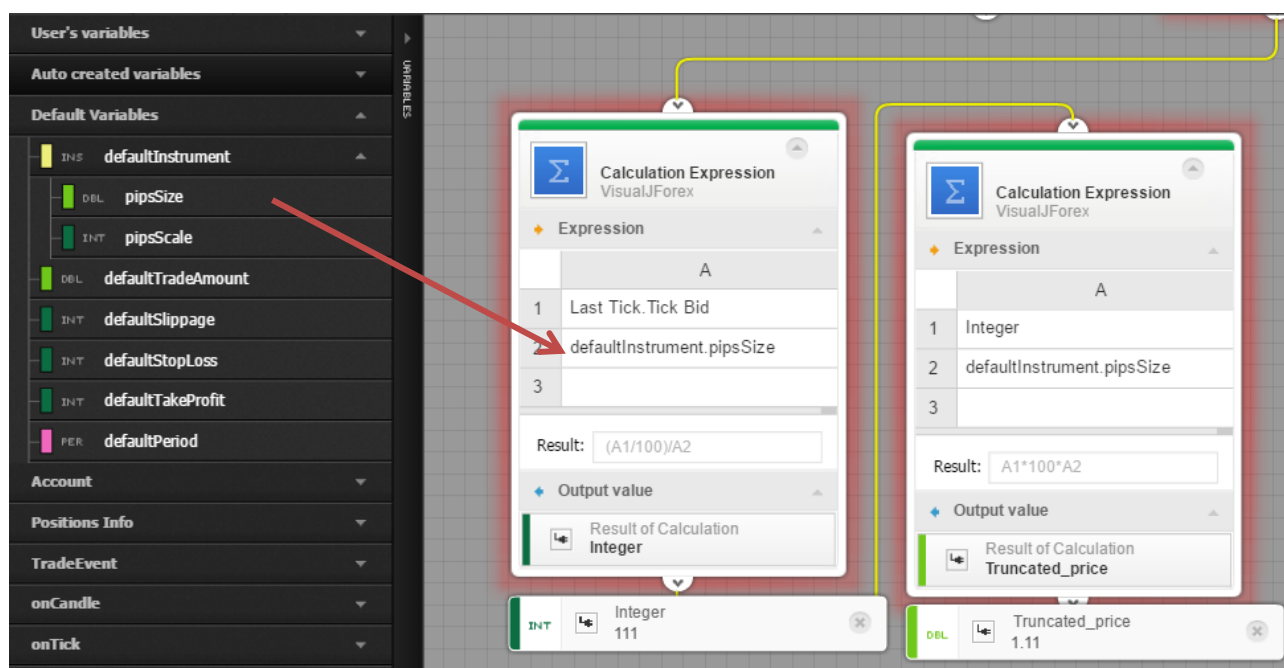
在右侧: 在该例子中，‘Trade event’（交易时间）做为起始点来获取 “Position Closed”（已平仓位）的信息，策略将检查货币对并分配“True”值给触发器，由此，新的仓位可以在下一流程被建立。

当需求是基于 Start / Hold（开始/保持）逻辑的时候，实行逻辑型触发器是可以行的。用户需要精确的确定触发器什么时候会转换以及什么时候会重置为初始值。拥有 3 或 4 个序列的更为复杂的流程可以使用拥有“Double” 或者 “Integer”的变量。

7.9. 如何缩短数字

缩短数字是减少数字小数点的操作，其目的是为了迎合交易平台的价格要求或者简化计算结果。缩短数字并不是四舍五入。

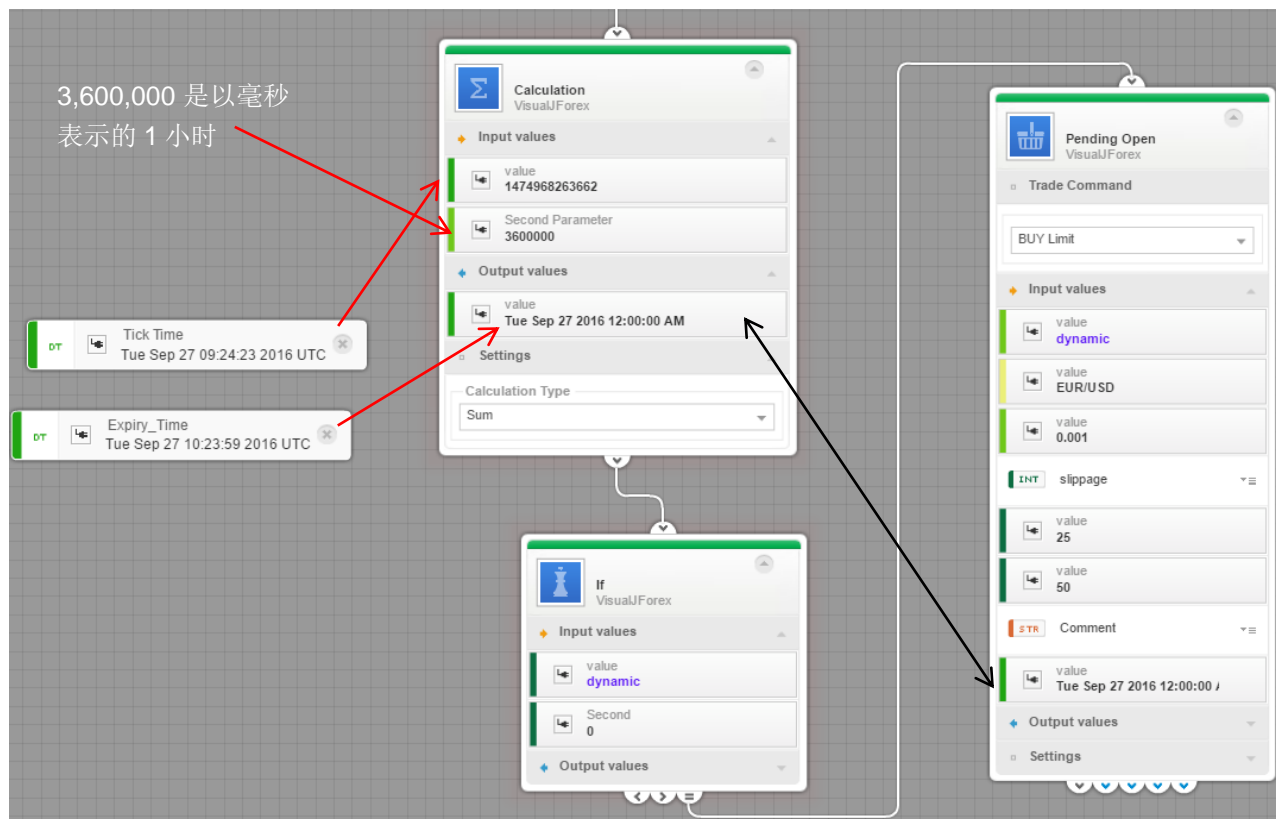
视像 JForex 并没有在现成的模块里应用此类功能，因此可以使用以下方法。



在这个例子中，上一个 tick 价格将被缩短至小数点后 2 位；第一步是获取 tick 价格乘以 100 然后保存在一个新的叫做“integer”的变量里。选择“integer”类型的目的是为了去除小数点。第二步是将结果除以 100，使其结果为 2 位小数点。“Pip size”（点值）变量被采用了，因为这个方法可以用到任何交易产品里。（例如：EUR/USD=0.0001 / USD/JPY= 0.01 / 股票和指数 =0.01）

7.10. 如何使用日期和时间

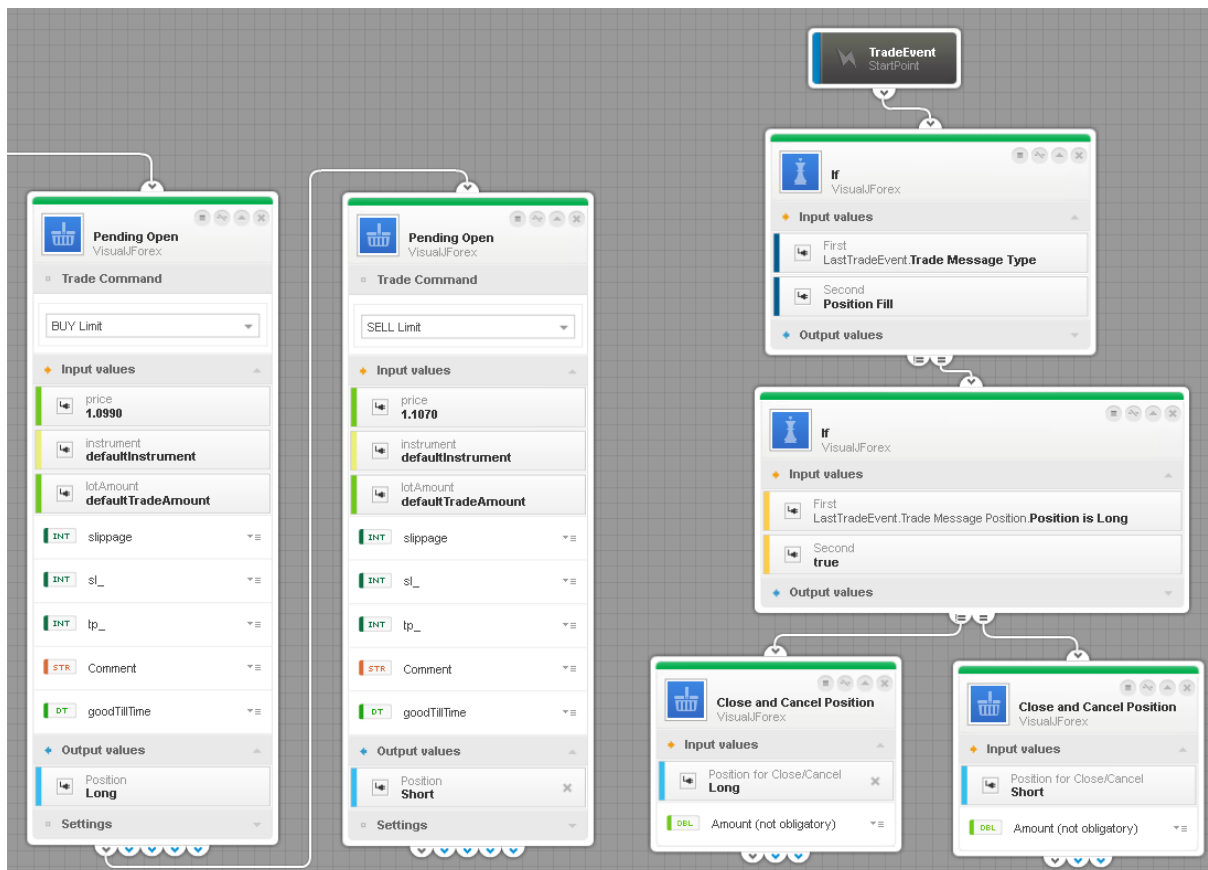
下面这个例子显示了如何在一个刚建立的限价单上添加一个到期时间。一旦达到入市条件，计算模块将捕获 tick 时间并在刚建立订单后决定订单到期时间。策略运行时，Tick 时间以时间格式显示在计算模块里，一旦变量建立在工作区里，它将被转换为一般的日期和时间格式。



7.11. 如果一个（多个）挂单已成立，如何取消剩余的挂单

当策略应用到挂单的时候，当一个挂单成交后，用户需要取消任何其它已建立的挂单。最好的方法便是和交易事件起始点配合使用，以此来获得”Fill”信息并发布一个取消仍处于挂单状态的剩余订单的请求。

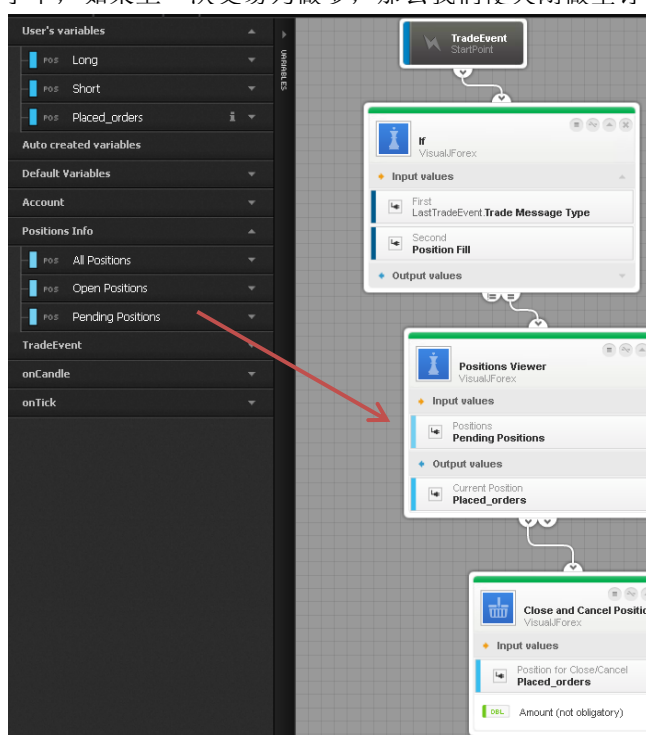
当策略处理有限数量的挂单的时候，您可以通过建立新的输出来鉴定订单，之后如果其中一个挂单被触发，策略将执行取消另一个订单的指令，如下：



从“Trade Event”（交易时间）起始点: 当收到交易信息类型“Order filled”（订单已成交）后，策略会核实仓位的方向并实行取消对立面订单的指令。在此例子中，如果上一次交易为做多，那么我们便关闭做空订单，反之亦然。

当策略处理多个已建立订单的时候，与“Position Viewer”（仓位观察器）模块配合使用将更为有效，这将对所有拥有“Placed”（已建立）状态的订单立即发送取消指令。

需要注意的是仓位观察器在“Pending Positions”（挂单）里作为输入值。



8. 错误信息和故障排除

8.1. 连接信息以及不完善的模块：



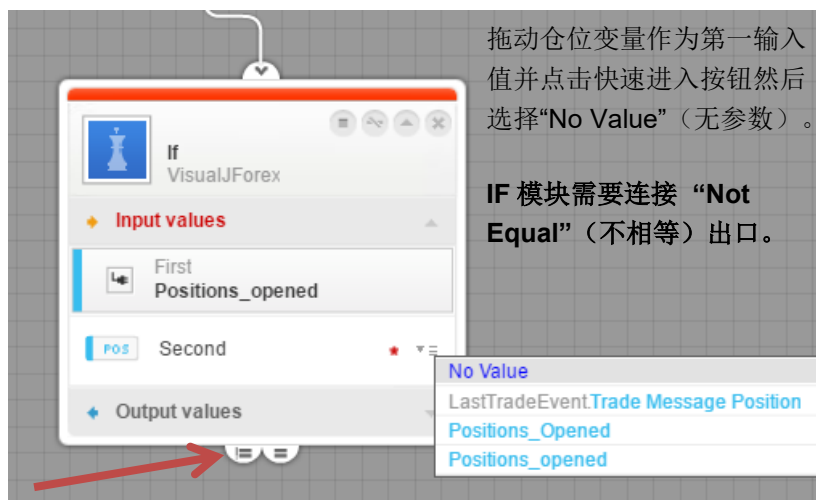
这个错误是当用户在某个模块里有遗漏参数的时候或者在没有与起始点互相建立链接的情况下尝试运行某个策略。它也会在没有策略文件打开的情况下，点击“Run”（运行）按钮后发生。

8.2. “No Value”（无值）错误



此类错误经常发生于当某个既定变量被使用的时候却没有给定相关参数。在上方错误信息里“Position_Opened.State”变量代表已开仓位的当前状态，无论其是否被成交。IF 条件尝试检查那些还没有被创建的仓位状态，因此得到此类错误反馈。为了避免此类情况的发生，用户需要多添加一个核实层级来确保“Opened positions”（已开仓位）变量在检查其状态前已被更新。

建立方法如下：



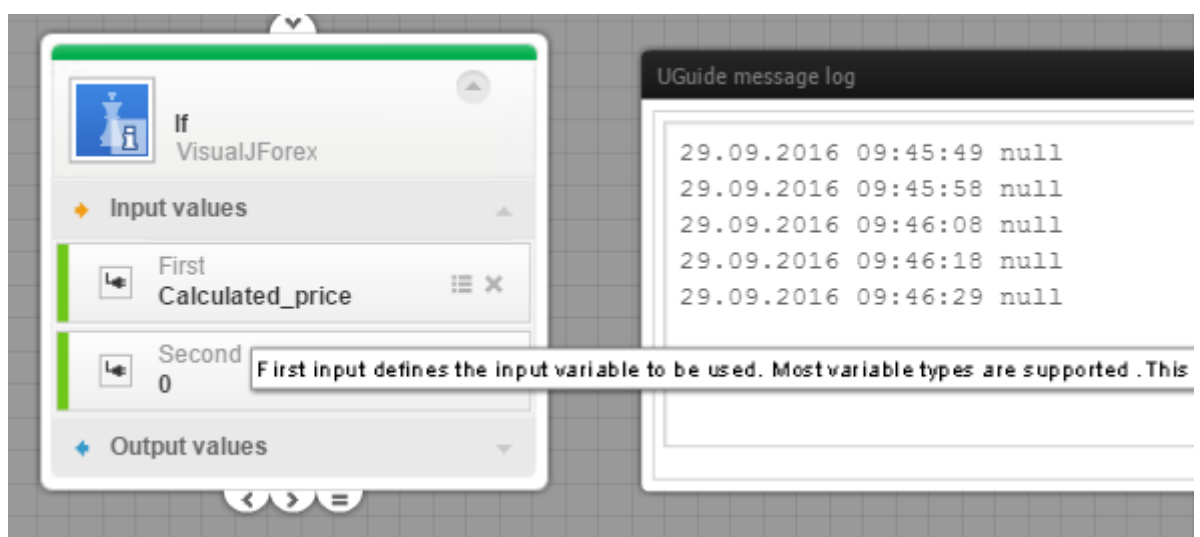
8.3. 变量命名：

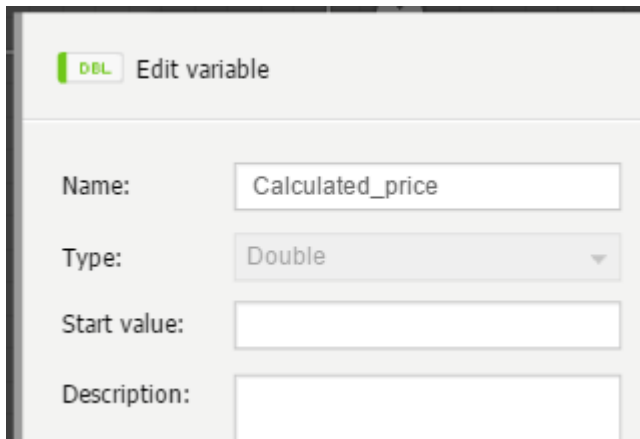


这个错误是关于变量命名格式的，命名不支持空格、符号并且不能以数字开头。

8.4. 变量缺失一个“start value”（起始值）

在章节 **Error! Reference source not found.** 中提到的关于变量创建和设置，下列错误通常由在创建变量的时候缺失属性：起始值。

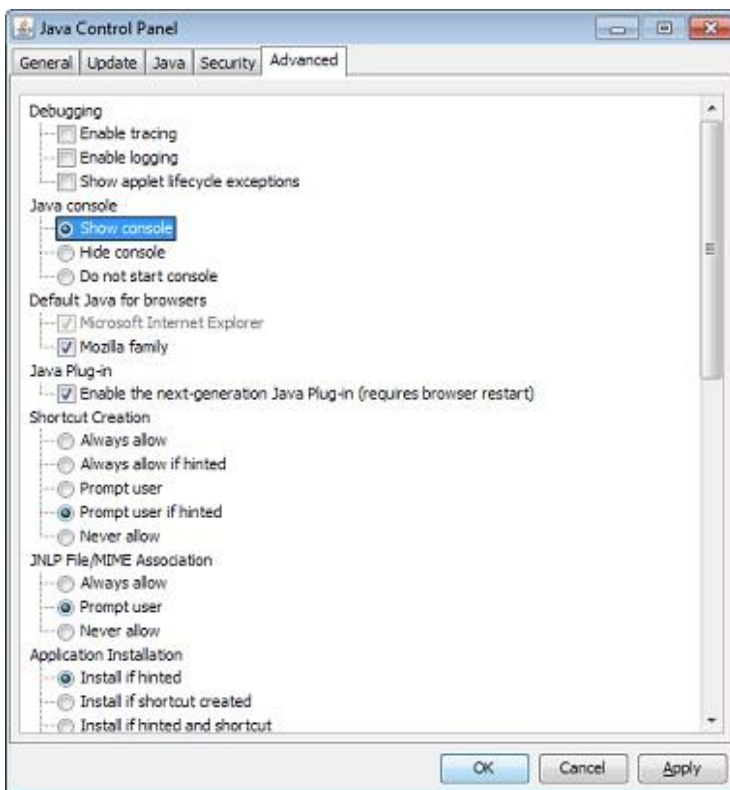




起始值是在创建的时候分配给变量的首个参数。变量“Calculated_price”（计算价格）的创建基于一个未被定义的起始值，系统会认定此变量拥有“No_Value”（无参数），因此任何逻辑型或计算型操作都将变为不可能。在某些例子中（类似于 8.2 中描述的错误），一个动态的变量可以在其还没有收到参数的时候产生此类错误，但一旦自动更新后（例如仓位状态），错误将不会再出现。

8.5. 高级调试

在此处，用户可以激活 Java 控制器来显示 Java 错误信息，这将帮助鉴别错误原因。该选项由 Java 控制面板激活。



当视像 JForex 处理器在运行的时候，将出现一个日志窗口，并记录任何与进程相关的追踪级别设定的信息。

拥有标题“java.lang.NullPointerException”的错误可能会影响策略的功能，因此需要被彻底调查。

进一步的信息和支持可以在此处查看（视像 JForex 论坛）[Visual JForex forum](http://visualjforex.com).