一、会签：

引擎中的会签即当前运行节点有多人进行办理，或者可以说当前节点存在多个办理任务。这些任务可以分为两种情况来运行，一种是按顺序进行办理，即串行会签；一种是同时进行办理，即并行会签。串行会签和并行会签都可以进行通过率的设置，串行会签中通过率是按照通过人数进行设置，达到通过人数则该节点办理完成，进入下一个节点。并行会签中通过率是按照人数的百分比进行设置，达到办理人数占总人数的百分比后，该节点办理完成。下面通过例子来对会签节点进行说明。

1. 流程定义中

<Activity name="任务" id="bb035582-9b73-460a-eaa5-bb3a786f9d3e" code="">

<Description>任务</Description>

<ActivityType type="TaskNode" complexType="SignTogether" mergeType="Parallel" completeOrder="2" compareType=”count”/>

……..

</Activity>

说明：

complexType：会签SignTogether或者加签SignForward模式

mergeType：并行Parallel或者串行Sequence模式

compareType：通过人数count或者百分比percentage

completeOrder：通过率，可以是通过率或百分比，由compareType决定

1. 流程转交时对下一步人员进行赋值
2. 为runner的NextActivityPerformers赋值。

接收人Performers的JSON格式为：{“ActivityGUID”,[{“UserID”:”UserName”}]}。

假设下一步人员有两个，那么以上格式可替换为：{“072af8c3-482a-4b1c-890b-685ce2fcc75d”,[{“01”:”张三”},{“02”:”李四”}]}

将以上结果转换为字典Dictionary<string, PerformerList>并赋值给WfAppRunner对象的NextActivityPerformers属性。例如：

PerformerList p1 = new PerformerList{UserID=”01”,UserName=”张三”};

PerformerList p2 = new PerformerList{UserID=”02”,UserName=”李四”};

Dictionary<string, PerformerList> dict = new Dictionary<string, PerformerList>();

dict.Add(“072af8c3-482a-4b1c-890b-685ce2fcc75d”,p1);

dict.Add(“072af8c3-482a-4b1c-890b-685ce2fcc75d”,p2);

WfAppRunner runner = new WfAppRunner();

runner.NextActivityPerformers= dict;

（2）调用引擎RunProcessApp(WfAppRunner runner)方法送往下一步

串行会签时引擎会按顺序生成办理任务，未办理的人员为挂起状态，待上一步人员办理完成后修改为办理状态，依次类推。

并行会签时引擎生成多人的办理任务，并且状态都为未接收状态，所有办理人员可以同时办理。

二、加签

引擎中的加签即当前办理节点中需要另外办理人加入时的操作。加签的方式分为三种：前加签、后加签和并行加签。

前加签：即新加入的办理人在发起加签的办理人之前办理。

后加签：即新加入的办理人在发起加签的办理人之后办理。

并行加签：即新加入的办理人和发起加签的办理人同时办理。

前加签和后加签均属于串行加签，按顺序进行办理。在加签中仍然可以设置通过率，由于加签是动态生成任务和节点，所以加签的通过率也需要动态传入。串行加签和并行加签的通过率和会签通过率的设置方式相同。以下为代码示例。

1. 流程定义

<Activity name="任务" id="bb035582-9b73-460a-eaa5-bb3a786f9d3e" code="">

<Description>任务</Description>

<ActivityType type="TaskNode" complextype="SignForward" />

……..

</Activity>

1. 发起加签

（1）选择加签人员

当该节点处于运行状态时，可以选择加签人员并设置加签通过率进行发起加签。

选择加签人员后，对WfAppRunner对象的NextActivityPerformers属性进行赋值，格式同会签格式。

Runner中DynamicVariables为动态变量，可将通过率赋值给它，比如：

DynamicVariables: {

"SignForwardType":SignForwardBefore，

“SignForwardCompleteOrder”:2

}

参数说明：

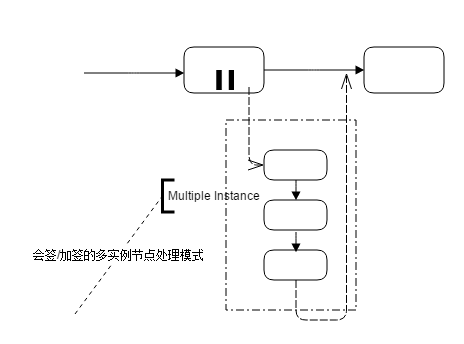
SignForwardType：加签通过类型，可分为SignForwardBefore前加签、SignForwardBehind后加签、SignForwardParallel并行加签

SignForwardCompleteOrder：加签通过率，串行加签为通过人数，并行加签为通过百分比

（2）调用引擎SignForwardProcess(WfAppRunner runner)发起加签

WfExecutedResult runAppResult = \_workflowService.SignForwardProcess(runner);

会签加签处理图形示意：



三、事件接口

通过在工作流转中需要调用外部事件来满足某些业务需求，所以引擎提供的事件接口。每个节点均可以绑定外部事件，包括开始和结束节点。关于外部事件的调用方法如下：

1. 流程定义中绑定事件接口

在Activity节点下增加Actions节点，如：

<Activity name="结束" id="725acc39-2ab7-44b6-ddd3-31f51a19864a" code="">

<Description>结束</Description>

<ActivityType type="EndNode" />

<Actions>

<Action type="ExternalMethod" name="UpdateStateThrough" assembly="Test.Methods" interface="Test.Methods.IService.InterfaceName" method="InterfaceMethod" />

</Actions>

………

</Activity>

Action节点属性说明：

type: 分为外部方法ExternalMethod和WebApi接口WebApi

name:自定义的调用名称用来标识action

assembly:调用外部方法所在的程序集名称

interface:调用的外部接口类名称

method:调用的外部接口中方法名称

外部程序集放在应用程序根目录下的\plugin目录中。

1. 流程转交时传入Action名称及参数

在流程转交时如果方法需要传参数，需要对runner的ActionMethodParameters属性赋值即可。如：

（1）实例化ActionParameterInternal参数类并赋值

var actionParameterInternal = new ActionParameterInternal();

actionParameterInternal.MethodParameters = new object[] { “abcd” };

（2）将参数赋给runner.ActionMethodParameters

runner.ActionMethodParameters = new Dictionary<string, ActionParameterInternal>();

runner.ActionMethodParameters.Add("UpdateStateThrough", actionParameterInternal);

（3）调用引擎RunProcessApp(WfAppRunner runner)方法送往下一步，引擎会自动调用外部事件并传入相关参数。