Agent Conversion Technical Reference

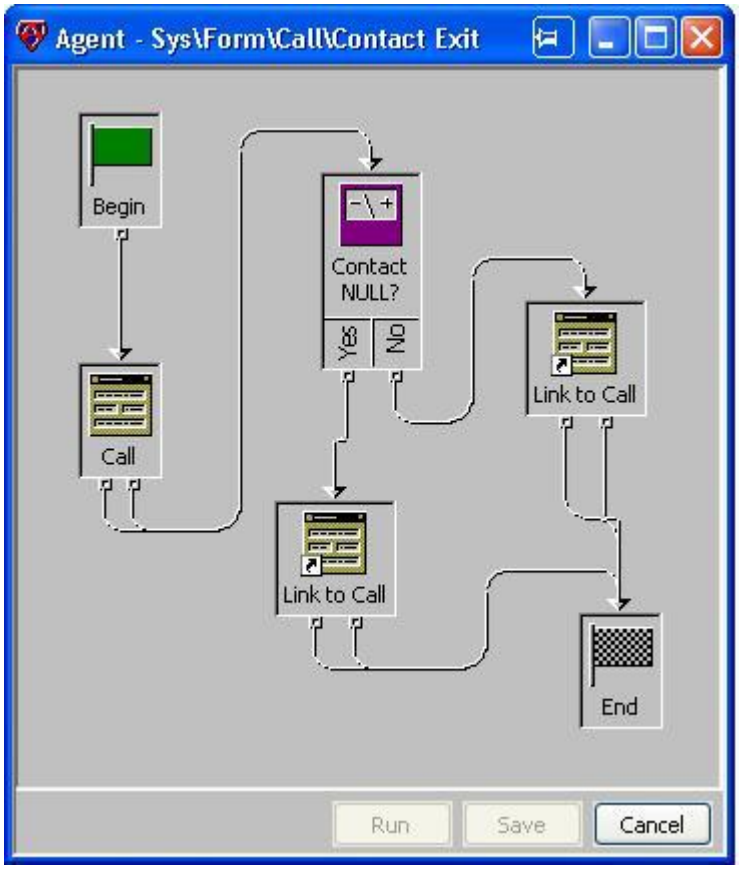
Oct-09

Table of Contents

Pivotal Agents to Client task – How it works

Pivotal Agent Conversion tool helps in converting the business workflows in Pivotal agents to Pivotal 6.0 supported Client Tasks. This is done by converting each agent into a c# class that can be then called inside any client task. Let us take a sample pivotal agent and see how it has been converted into a c# class that follows the entire agent logic.

The sample agent in the below example is "Call\Contact Exit"



When the above agent is converted, it creates one class and two .cs files, namely –

Sys_Form_Call_Contact_Exit.ComponentDefinition.cs

Sys_Form_Call_Contact_Exit.cs

Sys_Form_Call_Contact_Exit.ComponentDefinition.cs **– This file handles all the initialization of the agent. Hence, it contains the declaration of all the agent components and variables.**

Sys_Form_Call_Contact_Exit.cs – **This file enables running the Pivotal Workflow.**

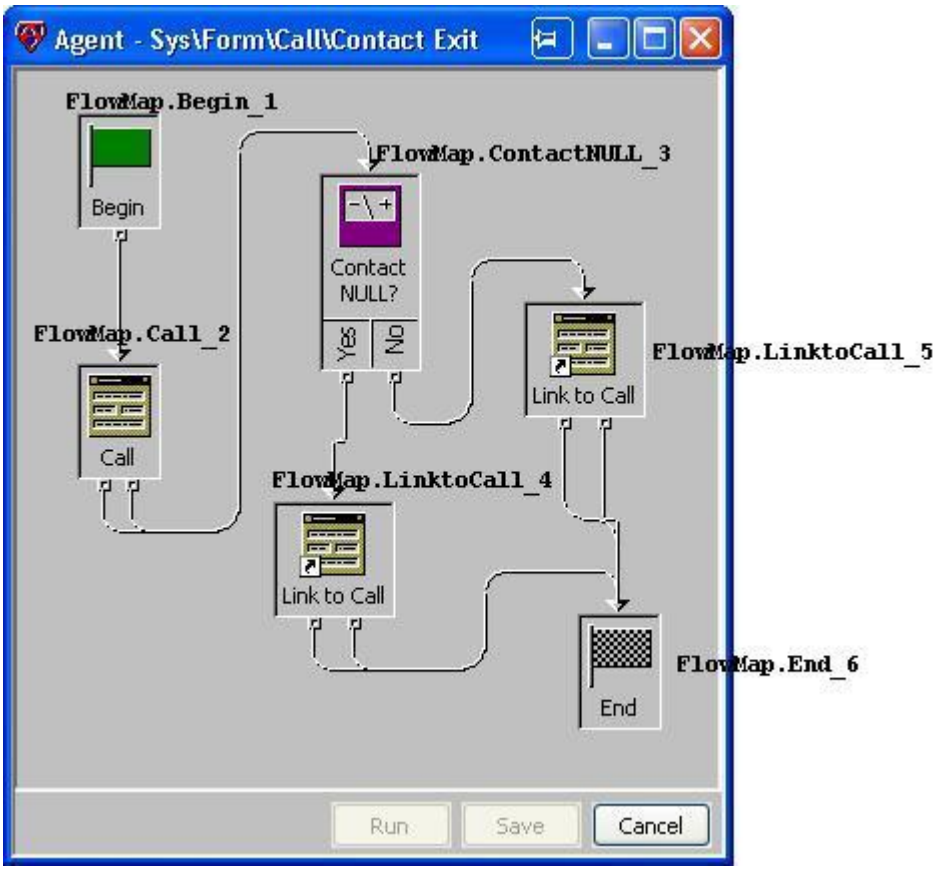
These two files together capture the two main parts in an agent –

1. **Flow of an agent –That is, where should the control go when steps in one component is finished.**
2. **Logic of the component –That is, the steps that need to be executed when control reaches a particular component.**

Flow of an Agent

To capture the flow of the agent, we need to track the number of components in the agent. This has been done in the <agentName>.ComponentDefinition.cs file as an enum FlowMap as shown below:

```
namespace CdcSoftware.Pivotal.Applications.AgentConversion.SharedAssemblies.DemoCT
{
    public partial class Sys_Form_Call_ContactExit
    {

        #region enum definition
        //This enum is used to map the flow of the agent between various components
        private enum FlowMap
        {
            Begin_1,
            Call_2,
            ContactNULL__3,
            LinktoCall_4,
            LinktoCall_5,
            End_6
        }
        #endregion
```



The names of the components have been modified so that they form valid enum values in c#. Hence a name like “Contact NULL?” would change to ContactNULL_3. Also agents support multiple components to have the same name, but this is not allowed in c#. Hence, we append a number to the end of the component name. Hence the two shortcuts “Link to Call” have been changed to LinktoCall_4 and LinktoCall_5 to differentiate them in the code.

Now, when an agent executes, the equivalent function that gets executed in the generated class is “Execute()”. The execute method for this agent class looks as shown below –

```
public void Execute()
{
    //The agent flow always starts with begin component

    m_flowMap = FlowMap.Begin_1;

    while (m_flowMap != FlowMap.End_6)
    {
        switch (m_flowMap)
        {

            case FlowMap.Begin_1:
                m_flowMap = DoBegin_1ComponentSteps();
                break;
            case FlowMap.Call_2:
                m_flowMap = DoCall_2ComponentSteps();
                break;
            case FlowMap.ContactNULL__3:
                m_flowMap = DoContactNULL__3ComponentSteps();
                break;
            case FlowMap.LinktoCall_4:
                m_flowMap = DoLinktoCall_4ComponentSteps();
                break;
            case FlowMap.LinktoCall_5:
                m_flowMap = DoLinktoCall_5ComponentSteps();
                break;
            case FlowMap.End_6:
                m_flowMap = DoEnd_6ComponentSteps();
                break;


        }
    }

}
```

In an agent, the flow always begins from “Begin” component and ends with “End” component. Hence, we first set the member m_flowMap to Begin as follows

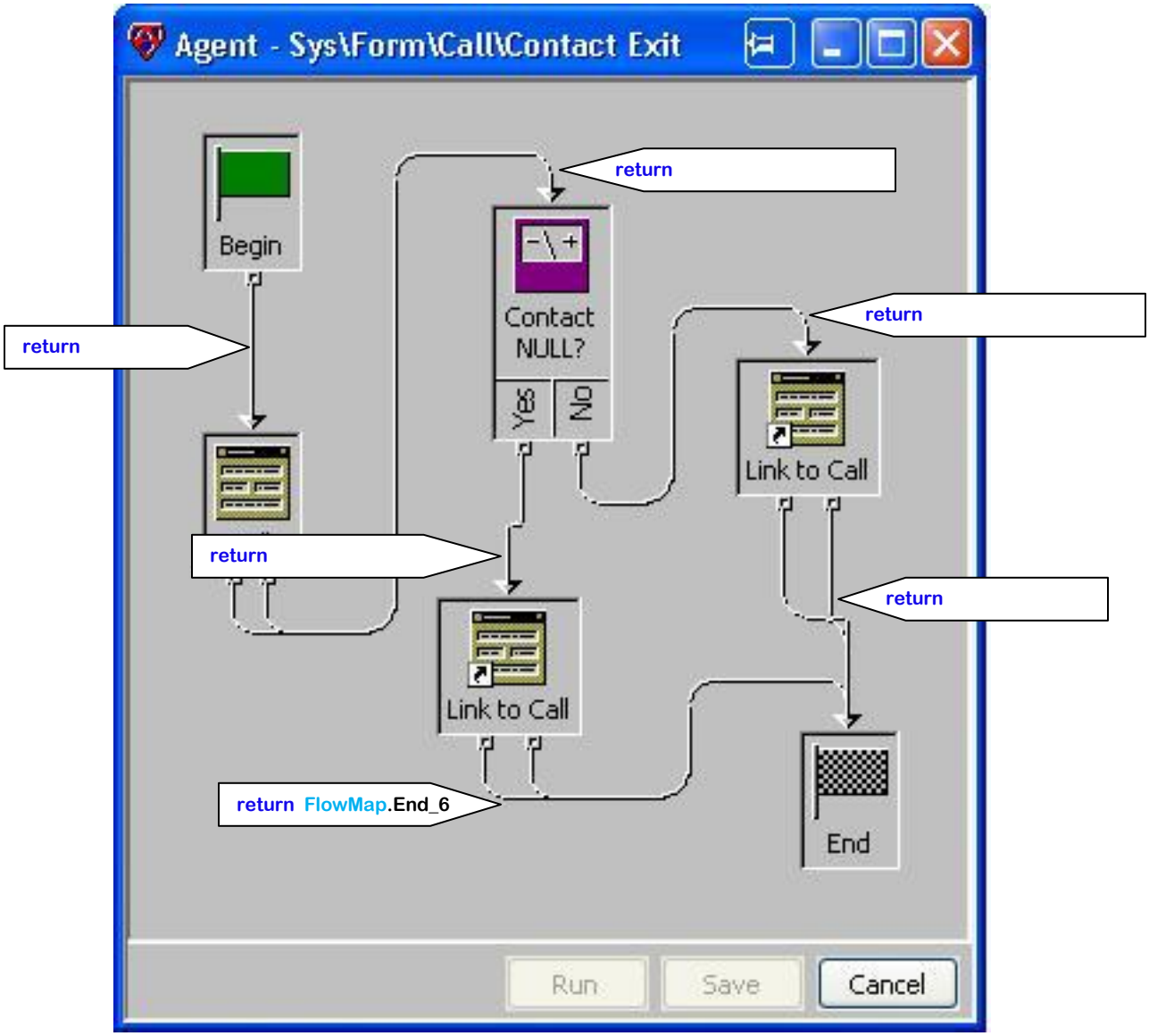
m_flowMap = FlowMap.Begin_1;

The rest of the flow of the agent depends on whether the steps present in the component executed successfully or not. Hence, in the generated code, the responsibility of providing the next component to go to lies with the component that has the control. A switch case is used to move from one control to another which depicts the flow in the agent.

Component Exit Paths

The control enters the switch case and calls Do_Begin_1ComponentSteps(); Since Begin component does not have any steps, this function usually contains only one statement return FlowMap.<next component enum>. Hence, in the sample agent it would be return FlowMap.Call_2 which represents the line connecting the Begin and Call components. Similarly, all the lines interconnecting the various components are represent by return statements in various "DoComponentStep" methods

Hence, the exit paths of all the components are handled by various "return" statements as follows -



Multiple Exit Paths

In cases of components where multiple exit paths are possible, the DoComponent functions take care of returning different enums. For ex: in case of "contact NULL?" in the example above, the DoContactNULL__3ComponentSteps looks as follows:

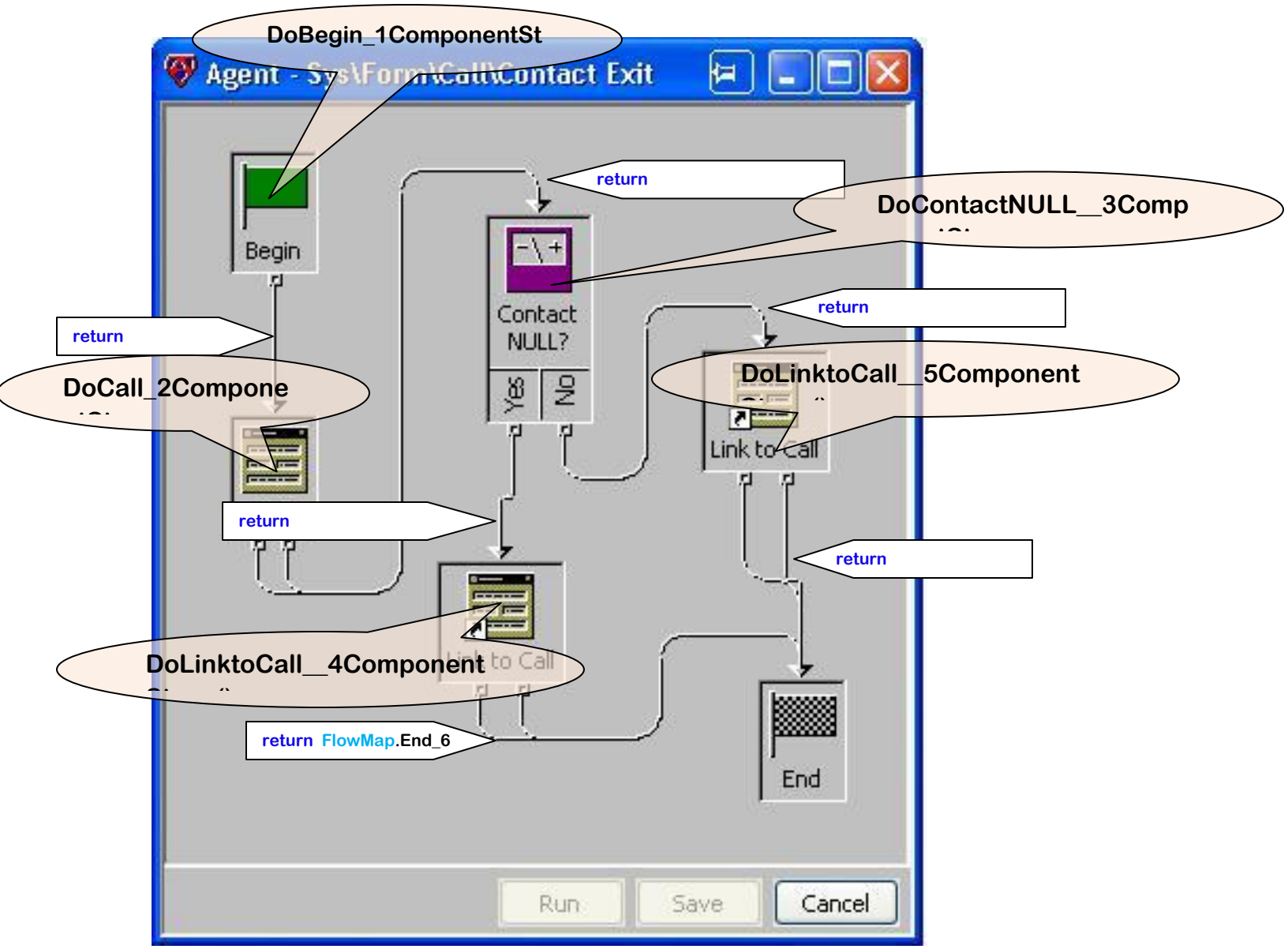
```
private FlowMap DoContactNULL__3ComponentSteps()
{
    try
    {
        if ((bool)FLogical.If((Id)MoveHelper.GetSourceComponentValue(HelperClass.GetComponent(m
            return FlowMap.LinktoCall_4;
        else
            return FlowMap.LinktoCall_5;
    }
    catch (Exception exc)
    {
        if (HelperClass.HandleException(exc) == 1)
            return FlowMap.End_6;
        else
            return FlowMap.End_6;
    }
}
```

For the moment, we will ignore the details of the above function. The function returns FlowMap.LinktoCall_4 when a particular condition is true and FlowMap.LinktoCall_5 when the condition is false which corresponds to the agent flow correctly.

When the flow of the component reaches the End component, it breaks the while loop condition and exits the "Execute" function. Care has been taken to connect unconnected links in an agent to End component to enable proper functioning of the generated code.

Logic of the Components

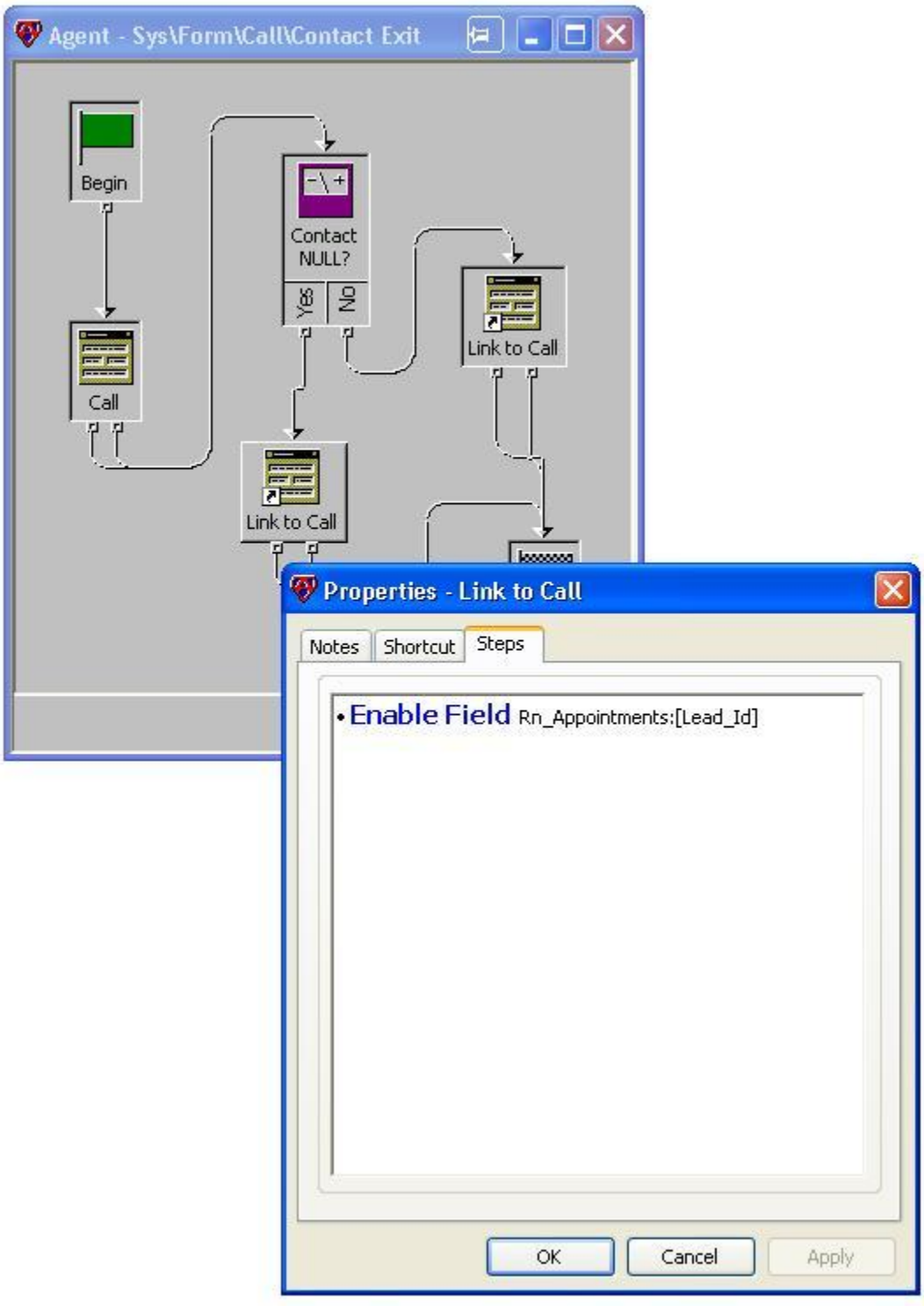
The logic handled in each of the components in an agent has been segregated into separate functions in the generated code as "DoComponentSteps" functions. Hence, the agent logic can be divided into agent component logic as follows –



Component Steps

The component steps are handled in each of the component's DoComponentStep functions.

For example: Link to Call component has one step – to enable a particular field



The corresponding DOCompoennt step code looks as shown below –

```
private FlowMap DoLinktoCall_4ComponentSteps()
{
    try
    {

      call_2.EnableField("Rn_Appointments", "Lead_Id");
      return FlowMap.End_6;

    }

    catch (Exception exc)
    {
        if (HelperClass.HandleException(exc) == 1)
            return FlowMap.End_6;
        else
            return FlowMap.End_6;
    }
}
```

Automatic Control Hookups

Form control hookup will be automatically done by the tool. When the WC Form is converted to a Smart Client Form, the Control Hookups will be automatically populated for the client form controls.

Once customizer builds the client task assembly, they should associate the assembly to the converted client form and all the control handlers would automatically be hooked to the respective control events.

These event hookups are available for a form.

Form Hookup

Form Hookups are the IFormClientTask events that require to perform basic operations on forms (for e.g. add, modify, delete etc)

WC Form event handlers (agent) would be added to the basic IFormClientTask methods in the generated code.

For e.g., an agent ***Sys\|Form\|Company\|OnSave(Modify)*** is being hooked to Form save event in the WC form. After conversion, the form hook up code will be as below in the Form Save Record method of the client task.

```
public override bool SaveRecord()
{
    try
    {
        if (this.FormData.RecordId == null)
        {
            return this.AddRecord();
        }
        else
        {
            Sys_Form_Contact_OnSave_Modify agent = new Sys_Form_Contact_OnSave_Modify();
            agent.FormAction = FormAction.Save;
            //Initializing Agent Components
            agent.InitializeAgentComponents();
            //Executing Agent Components
            agent.Execute();
            if (agent.PendingAction == PendingAction.None)
            {
                return base.SaveRecord();
            }
            else if (agent.PendingAction == PendingAction.Cancel)
                return false;
            else
                return true;

        }
    }
    catch (AgentStopException)
    {
        return false;
    }
    catch (Exception exc)
    {
        return Globals.HandleException(exc, true);
    }
}
```

Control Hookup

Event hookup for the control will be automatically hooked to the client task event handlers. For e.g. the Exit event on Phone Field will get converted as below:

```
public void Phone_OnExit(PivotalControl sender, EventArgs args)
{
    try
    {

        Sys_Form_Contact_Exit_Territory sys_Form_Contact_Exit_Territory = new Sys_Form_Contact_Exit_Territory();
        //Initializing Agent Components
        sys_Form_Contact_Exit_Territory.InitializeAgentComponents();
        //Executing Agent Components
        sys_Form_Contact_Exit_Territory.Execute();
    }
    catch (AgentStopException)
    {
    }
    catch (Exception exc)
    {
        Globals.HandleException(exc, true);
    }
}
```

Custom Bottom Bar Hookup

Custom Bar Buttons in a WC Form are treated as task pad items in a Client form. Hence, when these custom bar buttons get converted, the agents to which they are hooked are converted to command handlers, and these handlers need to be hooked manually. The code for button bar gets generated as:

```
[ClientTaskCommand]
public void Web_Detail_OnClick()
{
    try
    {
        Sys_Form_Contact_WebDetail sys_Form_Contact_WebDetail = new Sys_Form_Contact_WebDetail();
        //Initializing Agent Components
        sys_Form_Contact_WebDetail.InitializeAgentComponents();
        //Executing Agent Components
        sys_Form_Contact_WebDetail.Execute();
    }
    catch (AgentStopException)
    {
    }
    catch (Exception exc)
    {
        Globals.HandleException(exc, true);
    }
}
```

These steps have to be followed to associate client task methods:

- **Open the Task pad created for the converted form.**
- **Open each Task pad items for the task pad.**
- **Open command for each Task pad item.**
- **Associate the client task and command handler method to this command.**

- Save the task pad item.

Embedded Agent Workflow

Embedded agent workflow is used to encapsulate function body that can use all its parent objects. When the tool converts this embedded agent in to classes, its parent object collection gets passed through embedded class object property. Also the tool prefixes the embedded agent class name with its parent's agent name to bring more clarity.

Parent Class object will instantiate embedded class object with this way:

```
private FlowMap DoSetAccntMngr_11ComponentSteps()
{
    try
    {
        // Embedded Agent Call
        Sys_Form_Contact_Exit_Territory_SetAccntMngr setAccntMngr_11 = new Sys_Form_Contact_Exit_Territory_SetAccntMngr();
        //Setting component collection to Embedded agent class
        setAccntMngr_11.ComponentCollection = new Dictionary<string, IHelper>();
        foreach (KeyValuePair<string, IHelper> item in m_components)
        {
            setAccntMngr_11.ComponentCollection.Add(item.Key, item.Value);
        }
        //Setting variable collection to Embedded agent class
        setAccntMngr_11.VariableCollection = new Dictionary<string, VariableHelper>();
        foreach (KeyValuePair<string, VariableHelper> item in m_variables)
        {
            setAccntMngr_11.VariableCollection.Add(item.Key, item.Value);
        }
        //Initializing Agent Components
        setAccntMngr_11.InitializeAgentComponents();
        //Executing Agent Components
        setAccntMngr_11.Execute();
        return FlowMap.ContactHasOpportunities_12;
    }
    catch (PivotalApplicationBaseException exc)
    {
        if (HelperClass.HandleException(exc) == 1)
            return FlowMap.ContactHasOpportunities_12;
        else
            throw new AgentStopException("Agent Execution Stopped");
    }
}
```

Embedded Class will contain the object collection property as:

```
internal override Dictionary<string, IHelper> ComponentCollection
{
    get
    {
        return m_components;
    }
    set
    {
        m_components = value;
    }
}

internal override Dictionary<string, VariableHelper> VariableCollection
{
    get
    {
        return m_variables;
    }
    set
    {
        m_variables = value;
    }
}
```

Call Agent Workflow

Call agent workflow works as common functionality reused for different business workflow. These Call agents get converted as different class of different or same assembly as selected by the customer.

```
private FlowMap DoExit_Territory_16ComponentSteps()
{
    try
    {
        //TODO: Do the following steps before using call agent
        //  A.  Build the shared assembly containing call agent class logic
        //  B.  Add that shared assembly reference manually
        //  C.  Include the assembly namespace with using directive to use agent class


        Sys_Form_Contact_Exit_Territory sys_Form_Contact_Exit_Territory = new Sys_Form_Contact_Exi
        sys_Form_Contact_Exit_Territory.FormAction = this.FormAction;
        sys_Form_Contact_Exit_Territory.PendingAction = this.PendingAction;
        sys_Form_Contact_Exit_Territory.BeginParamList = exit_Territory_16.BeginParams;
        //Initializing Agent Components
        sys_Form_Contact_Exit_Territory.InitializeAgentComponents();
        //Executing Agent Components
        sys_Form_Contact_Exit_Territory.Execute();
        exit_Territory_16.EndParams = sys_Form_Contact_Exit_Territory.EndParamList;

        return FlowMap.End_15;
    }
    catch (PivotalApplicationBaseException exc)
    {
        if (HelperClass.HandleException(exc) == 1)
            return FlowMap.End_15;
        else
            throw new AgentStopException("Agent Execution Stopped");
    }
}
```

Navigation Client Task

Windows Client has agents that can be run on timely basis to perform certain task (For e.g. Data Admin agents). The tool provides an option to convert these agents as Navigation Client Task such that they can be hooked to navigation items.

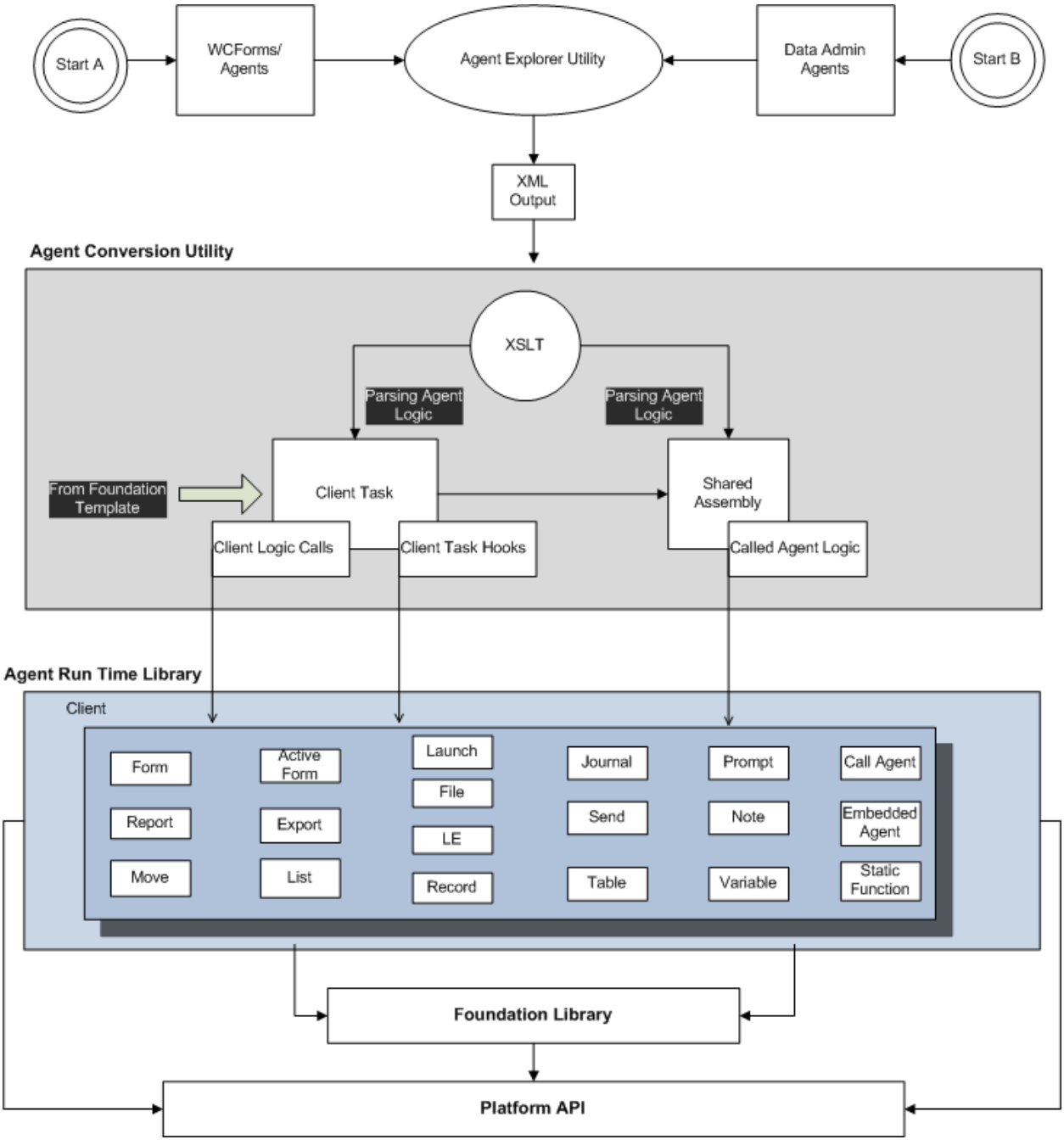
For e.g., the tool will generate navigation client task code for "Currency Dup Checking" agent as:

```
[ClientTaskCommand]
public virtual void Execute()
{

    //Initializing Agent Components
    this.InitializeAgentComponents();

    //The agent flow always starts with begin component

    m_flowMap = FlowMap.Begin_1;
                enum CdcSoftware.Pivotal.Applications.AgentConversion.NavigationClientTasks.PurgeTickleRecord.RecordTickling_Purg
    while (m_flowMap != FlowMap.End_5)
    {
        switch (m_flowMap)
        {

            case FlowMap.Begin_1:
                m_flowMap = DoBegin_1ComponentSteps();
                break;
            case FlowMap.SetParameters_2:
                m_flowMap = DoSetParameters_2ComponentSteps();
                break;
            case FlowMap.SysTicking_3:
                m_flowMap = DoSysTicking_3ComponentSteps();
                break;
            case FlowMap.SysTicking2_4:
                m_flowMap = DoSysTicking2_4ComponentSteps();
                break;
            case FlowMap.End_5:
                m_flowMap = DoEnd_5ComponentSteps();
                break;


        }
    }

}
```

These steps have to be followed to use the these agents workflow in client application

- **Create Navigation Client Task by selecting agent from Toolkit**
- **Build the Generated Navigation Client task and import the code file to the system**
- **Create a separate Navigation items (such as Subject, Topic, Task Pad item) to hook the Client task**
- **Create a command of type client task and associate the generated client task**
- **Associate Command Handler to Execute Method of Client Task**
- **Save the Navigation Item and give the security permission to access in Client application**

Architecture Diagram



Agent Component Mapping

Introduction

This document will provide details about Pivotal Agent Component mapping to Agent Run Time API's.

Record Component

Class Name: RecordHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client
Properties

Name	Type	Description	Pivotal Agent Property
ActivityLogId	Id	Gets the journal page id for the current record	Activity Log Id
DataRow	DataRow	Gets the current data row for record component	Similar to return reference row from record component methods: Next Record New Record
DataTable	DataTable	Gets the current data table loaded by record component of either object, static list or query	Complete record set return from Record component
LastVerifyError	String	Gets the Error message when a record is tested for formula field using VerifyRecord method	Last Verify Error
Name	String	Gets the Name of the component	Name of the component
ObjectName	String	Gets the Name of the Object used for the	Name of the Object used for the

		component	component
ParameterCollection	ParameterList	Gets/Sets the Query Parameter Collection	Param collection for open query used in record component
RecordCursor	Int	Gets the Current Cursor index on the record set of record component	Current cursor that gets updated for each Next Record call
RecordId	Id	Gets the Record id of the current record	Record Id
TableId	Id	Gets the Table Id of the current record object	Table Id

Method

Name	Parameter	Description	Pivotal Agent Step
Activate	Record id (Type: object)	Opens the interaction record in outlook if the underneath record is on Rn_Interactions table	Activate method that used to open record of Rn_Appointments in calendar view
AddToQueryParameter	Parameter for query (Type: object)	Adds the parameters to open parameter query to build record set	Query Parameter Param Strings
DeleteRecord	None	Delete Current record	Delete Record
NewRecord	None	Create a new record	New Record
NextRecord	None	Move the cursor to next row	Next Record

ReadRecord	Record Id (Type: object)	Set the record id to load the record in record component	Read Record
Refreshrecordfrom Database	None	Gets the current record from database by cancelling all changes in the cache	Refresh Record from Database
ReleaseRecord	None	Release record that is being changed in the workflow	Release Record
Reset	None	Clears off the record cursor to null and loads the complete recordset again	Reset
SaveRecordtoData base	None	Save the record	Save Record to database
TickleRecord	None	Send the tickle indication to PBS for mobile record	Tickle Record
VerifyRecord	Record Id (Type: object)	Verify the record for formula fields	Verify Record
Move	5 Overloaded methods to move data as: Variable to Record Property Record Property to Variable Component to Record Property Record Property to	General Move statement to move data from one source object to destination	Move Operation

	Component Static Value to Record Property		
Add	5 Overloaded methods to add data as: Variable to Record Property Record Property to Variable Component to Record Property Record Property to Component Static Value to Record Property	General Add statement to add data in destination object	Add Operation

Form Component

Class Name: FormHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
ActivityLogId	Id	Gets the journal page id associated to record of the form (if any)	Activity Log ID
FormId	Id	Gets the Form Id for the form component	Form Id
FormName	String	Gets/Sets the form name of form	

		component	
Name	String	Gets the Name of the component	Name of the component
ObjectName	String	Gets/Sets the Name of the Object used for the component	Name of the Object used for the component
RecordId	Id	Gets/Sets the record id of the form loaded	Record Id
SecondaryRecordId	Int	Gets the Current Cursor index record id in selected secondary	Secondary Record Id
SecondaryRecordCount	Int	Gets the count of rows in selected secondary	Secondary Row Count
SecondaryRowNumber	Int	Gets/Sets the Current Cursor index in selected secondary	Secondary Row Number
TableId	Id	Gets the Table Id of the current form object	Table Id

Method

Name	Parameter	Description	Pivotal Agent Step
Close	Boolean Flag to show warning (Type: Bool)	Close the form with/without warning as per flag	Close <warning>
ControlHandlerForCell	Sender Object (Type: Object) Argument (Type: EventArgs)	Populate secondary property on cell event handler	Populate secondary property

ControlHandlerFor Secondary	Sender Object (Type: Pivotal Control) Argument (Type: EventArgs)	Populate secondary property on row event handler	Populate secondary property
DisableField	Section Name (Type: string) Field Name (Type: String)	Disable a Field	Disable field <f>
EnableField	Section Name (Type: string) Field Name (Type: String)	Enable a Field	Enable field <f>
EnterField	Table Name (Type: string) Field Name (Type: String)	Opens a dialog window to populate field value a segment field	Enter field <f>
NewLineOnSecondary	Table Name (Type: string)	Opens a dialog window to populate new row to secondary	New Line on <t>
OpenForModify	None	Opens the form with form name and record id in center content	Open for Modify
OpenNew	None	Opens the form with form name in center content	Open New
ReadOnlyField	Section Name (Type: string) Field Name (Type: String) Read only flag (Type: Bool)	Set the field to Read Only as per flag	Read Only Field <TRUE|FALSE><f>

SetFocus	Section Name (Type: string) Field Name (Type: String)	Set the focus on the field	Set focus to <f>
Next	None	Load the next form record	Next
Previous	None	Load the previous record	Previous
LetterExpress	None	Load the letter express meta window where user can choose the item and execute	LetterExpress
Print	None	Print the record using Single <Object_Name> report	Print
Move	4 Overloaded methods to move data as: Variable to Component Component to Variable Component to Component Variable to Variable	General Move statement to move data from one source object to destination	Move Operation

Unsupported Form Component Steps/Properties

Name	Step/Property	Description	Manual Workaround
Activities	Step	Show the Log Page form for the record	Platform API not supported
Control Frame	Property	Id of the form used to control previous	Platform API not

Number		or next bar button on the form	supported
Control Grid ID	Property	Secondary Table Id within the form identified by the control frame number property	Platform API not supported
Position	Step	Set the list to x,y co-ordinate of window	Platform API not supported

Send Component

Class Name: SendHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
AddToLog	bool	Gets/Sets the value	Add to Log
AddToFaxLog	bool	Gets/Sets the value	Add to Fax log
Message	String	Gets/Sets the message	Message
Subject	String	Gets/Sets the email subject	Subject
MessageId	String	Gets the message ID	MessageId
Name	String	Name of the Send Component	

Method

Name	Parameter	Description	Pivotal Agent Step
AddRecipient	type (Type: RecipientType) emailAddress (Type: string) Name (Type: string)	Adds a recipient to the message	Add <type> Recipient <x> Named <y>
AddAttachment	filePath (Type: string)	Adds an attachment to the message	Add Attachment <x>
AddShortcut	type (Type: string) tableId (Type: Id) rowId (Type: Id) formId (Type: Id) name (Type: string)	Adds a shortcut to the message	Add <type> Shortcut for Table <t> Record <r> Form <f> Called <y>
Send		Send the Message	Send
MoveShortcut	shortcutName (Type: string) destObject (Type: Object)	Move Shortcut	Move Shortcut <text> to <f/p>
LogRelatedRecord	tableId (Type: Id) recordId (Type: Id)	Log related record to send communication	Log Record <x> for table <t>
Move	4 Overloaded methods to move data as: Variable to Component Component to Variable Component to Component	General Move statement to move data from one source object to destination	Move Operation

	Variable to Variable		

Launch Component

Class Name: LaunchHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
DefaultDirectory	string	Gets/Sets the Default Directory Path	Default Directory
LaunchString	string	Gets/Sets the Application Path	Launch String
CommandLineString	string	Gets/Sets Command Line Arguments	Parameters

Method

Name	Parameter	Description	Pivotal Agent Step
Application	None	Launch an application with specified command line arguments	Application
Document	None	Launch a document	Document
WebPage	None	Launch a Webpage	Web page

File Component

Class Name: FileHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
FileName	string	Gets/Sets the File Name/File Path	File Name
Name	String	Name of the File Component	

Method

Name	Parameter	Description	Pivotal Agent Step
LoadFile	moveType (Type : MoveType) destObject(Type: object) destProp(Type: string)	Load file to form field or record field	Load to <f/p/v>
LoadFile	destObject (Type: object) destProp (Type: string) secondaryTableName (Type: string)	Load file to a secondary table field	Load to <f/p/v>
LoadFile	binaryContent (Type : object) attachmentVariable (Type: VariableHelper)	Load file content to an attachment variable	Load to <f/p/v>
SaveFileToDisk	sourceObject (Type: object)	Save Attachment from Secondary to	Save <f/p/v> to Disk

	sourceProp (Type: string) secondaryTableName (Type: string) filePath (Type: string)	disk	
SaveFileToDisk	moveType (Type: MoveType) sourceObject (Type: object) sourceProp (Type: string) filePath (Type: string)	Save Form Field/Record Field Attachment To disk	Save <f/p/v> to Disk
SaveFileToDisk	attachmentVariable (Type: VariableHelper)	Save attachment variable content to disk	Save <f/p/v> to Disk
DeleteFile	moveType (Type: MoveType) sourceObject (Type: object) sourceProp (Type: string)	Delete Attachment from a from Field or Record Field	Delete

Export Component

Class Name: ExportHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
FileName	string	Gets/Sets the file path	File
Name	String	Get the Name of the Export Component	

Method

Name	Parameter	Description	Pivotal Agent Step
NewFile	None	Opens a file using the name specified. If a file of the same name already exists, it is overwritten with new data.	New File
OpenFile	None	Opens an existing file and appends data to it. If no file exists, it is created.	Open File
CloseFile	None	Closes the file specified	Close File
EndOfLine	None	Starts a new line in the specified file	End of Line
Write	2 overloaded method to take any source object value	Writes parameter <x> to the target file. Commas are inserted between fields on the same line.	Write <x>
Move	4 Overloaded methods to move data as: Variable to Component Component to Variable Component to Component Variable to Variable	General Move statement to move data from one source object to destination	Move Operation

Report Component

Class Name: ReportHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
PrinterAsDestination	bool	Gets/Sets the value	Design time Printer check box
DiskAsDestination	bool	Gets/Sets the value	Design time Disk File check box
WindowAsDestination	bool	Gets/Sets the value	Design time Windows check box
RecordId	ID	Gets/Sets the Record ID	Record Id
StaticListID	ID	Gets/Sets the Static List ID	Static List Id
ReportFileType	ReportFileType	Gets/Sets the value of ReportFileType	Type
FileName	String	Gets/Sets the value	File
Name	String	Name of Record Component	
ObjectName	String	Name of the Table Object	Table object
ParameterListCollection	Object	Parameter collection for open ended query	Param string
ReportName	String		
TableName	String		
QueryName	String	Query Name	Query Name

Method

Name	Parameter	Description	Pivotal Agent Step
Execute	None	Execute the report	

Letter Express Component

Class Name: LetterExpressHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
AddToLog	Bool		Add To Log
LogMergeDoucment	Bool		Log Merge Document
DestinationType	LEDestination		
LetterExpressName	string		
RecordId	Id		
UseRecordsFrom	LEUseRecordsFrom		
StaticListID	ID	Gets/Sets the Static List ID	Static List Id
Name	String	Name of Record Component	
ObjectName	String	Name of the StaticList/Query	StaticList/Query
ParameterListCollection	Object	Parameter collection for open ended query	Param string
LetterExpressName	String		
TableName	String		

Method

Name	Parameter	Description	Pivotal Agent Step
Execute	None	Execute the LE	

List Component

List helper shows the results set to the user to select the item. This results set can be based at design time or it may have item added at run time and hence the list can be divided into class objects:

- Based on Query/Table (Design Time)
- Based on Static List (Run Time)

List For Query/Table (Design Time)

Class Name: ListHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
CurrentlyHighlightedRecord	Id	Gets/Sets the currently highlighted record of the list	Current Highlighted Record
CurrentlyHighlightedText	String	Gets/Sets the currently highlighted record Rn_Descriptor	Currently Highlighted Text
Name	String	Gets the Name of the component	Name of the component
ObjectName	String	Gets the Name of the Object used for the component	Name of the Object used for the component
QueryName	String	Gets/Sets the Query Name if list of type Query	Query Name
StaticList	StaticList	Gets the Static List	

		object if the type is static List	
Static List Id	Id	Gets/Sets the static List Id	Static List Id
ParameterCollection	ParameterList	Gets the parameter collection if type is query	Param[1…]

Method

Name	Parameter	Description	Pivotal Agent Step
AddToQueryParameter	Parameter for query (Type: object)	Adds the parameters to open parameter query to build record set	Query Parameter Param Strings
Close	None	Close Static List	Close
HighlightFirstItem	None	Move List component focus to first row	Highlight First Item
HighlightLastItem	None	Move List component focus to last row	Highlight Last Item
HighlightNextItem	None	Move List component focus to next row	Highlight Next Item
HighlightPreviousItem	None	Move List component focus to previous row	Highlight Previous Item
HighlightRecord	Record Id (Type: Id)	Move List component focus to record id row	Highlight Record Id Item
Open	None	Open the static	Open

		list	
Move	5 Overloaded methods to move data as: Variable to Record Property Record Property to Variable Component to Record Property Record Property to Component Static Value to Record Property	General Move statement to move data from one source object to destination	Move Operation
SaveHighlighlighteddRecord	2 Overloaded method to store the highlighted record	Save Highlighted Record To variable or component object property	Save Highlighted Record to <x>
SaveHighlightedText	2 Overloaded method to store the highlighted record text	Save Highlighted Record Text To variable or component object property	Save Highlighted Text to <x>

Unsupported Pivotal Agent APIs

Name	Step/Property	Description	Manual Workaround
Position	Step	Set the list to x,y co-ordinate of window	Platform API not supported

List For Static List (Run Time)

Class Name: StaticListHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
StaticList	StaticList	Gets the Static List object if the type is static List	
Static List Id	Id	Gets/Sets the static List Id	Static List Id

Method

Name	Parameter	Description	Pivotal Agent Step
AddRecordToStaticList	Record id (Type: Id)	Add record to static list for static list type	Add <x> to Static List
ClearStaticList	None	Clear all records from static list	Clear Static List
CreateStaticList	Static List Name (Type: String)	Create static list with name. If it already exists in ED, it will load it	Create Static List <x>
DeleteStaticList	Static List Name (Type: String)	Delete Static List	Delete Static List <x>
RemoveRecordFromStaticList	Record Id (Type: Id)	Remove Record Id item from static list	Remove <x> from Static List

Active Form Component

Class Name: FormHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
ActiveParameter	ParameterList	Gets the transit point parameter collection	
DataSet	DataSet	Gets the dataset associated to active form	
DataTemplate	DataTemplate	Gets the data template for the active form	
Name	String	Gets the Name of the component	Name of the component
ObjectName	String	Gets the Name of the Active Form	Active Form Name
RecordId	Id	Gets/Sets the record id of the form loaded	Record Id
LastErrorText	String	Gets the error Text for last active form execution	Last Error Text
LastErrorCode	Int	Gets the error HRESULT Value for last active form execution	Last Error Code
ParamCount	Int	Gets the active parameter count	Active Parameter Count

Method

Name	Parameter	Description	Pivotal Agent Step
AddFormData	None	Add Data for Form Script method	Add Form Data
AddToActiveParameter	Param (Type: Object)	Add parameter to the active parameter collection starting from zero index. Any user defined parameter are passed from 6th index to active form method	Add <value> to Active Parameter
DeleteFormData	None	Delete Data for Form Script method	Delete Form Data
DeleteRecrdFromSecondary	Number (Type: int) secName (Type: String)	Delete the number of row from current cursor on the secondary record	Delete <n> Records From Secondary <secondary>
ExecuteMethod	Method Name (Type: string)	Execute a form method	Execute Method
LoadFormData	None	Load Data for a form script	Load Form Data
MoveFirstOnSecondary	Secondary Name Name (Type: string)	Move the cursor to first record of secondary	Move First on Secondary <secondary)
MoveLastOnSecondary	Secondary Name Name (Type: string)	Move the cursor to last record of secondary	Move Last on Secondary <secondary)
MoveNextOnSecondary	Secondary Name Name (Type: string)	Move the cursor to next record of secondary	Move Next on Secondary <secondary)

MovePreviousOnSecondary	Secondary Name Name (Type: string)	Move the cursor to previous record of secondary	Move Previous on Secondary <secondary)
MoveRecordOnSecondary	Number (Type: int) Secondary Name (Type: String)	Move the cursor to number of records of secondary	Move Records <n> on Secondary <secondary)
MoveParameterValue	Index (Type: int) Object (Type: Object>	2 overloaded method to move transit point parameter value to target object	Move <n> Active Parameter to <target>
MoveToField	Object (Type: Object) FieldName (Type: String)	3 overloaded method to move values to primary field	Move <target> To Field
MoveToSecondaryData	Object (Type: Object) Secondary Name (Type: String) FieldName (Type: String)	3 overloaded method to move values to secondary field	Move <target> to Secondary <secondary> field
NewFormData	None	New Data For Form script	New Form Data
NewSecondaryData	Secondary Name (Type: String)	New Secondary Data for the secondary	New Secondary Data on Secondary <secondary>
Reset	None	Reset the parameter	Reset Active Parameters
SaveFormData	None	Save Data for Form Script	Save Form Data

Journal Component

Class Name: JournalHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
ActivityLogId	Id	Get the journal page id for primary table and record id	
Name	String	Get the Name of the Export Component	Journal Component Name
RecordId	Id	Primary Record Id on to which journal item is based	
TableId	Id	Table Id table on which RecordId is based	

Method

Name	Parameter	Description	Pivotal Agent Step
GetJournalPage	tableId (Type: Id) recorded (Type:Id)	Gets the journal page related to primary record and table Id	Get Journal Page
AddJournalItem	tableId (Type: Id) recordId (Type:Id)	Add related linked item to base journal record.	Add journal Item

Note Component

Class Name: NoteHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
Name	String	Get the Name of the Export Component	Name of Note Component

Method

Name	Parameter	Description	Pivotal Agent Step
ShowMessage	text (Type: String) title (Text: String)	Opens a Dialog box with string text and title	

Prompt Component

Class Name: PromptHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
Name	String	Get the Name of the Export Component	Name of the prompt component
ButtonTexts	List<string>	List string array of button option text	
PromptStyle	PromptStyle	Enum to set the type of prompt	Command Buttons Or Radio Buttons
Text	String	Gets/Sets the Text property of Prompt component	Text
Title	String	Gets/Sets the Title property of Prompt	Title

		component	

Method

Name	Parameter	Description	Pivotal Agent Step
AddOptionsToPrompt	option (Type: String)	Add the command button/radio button options	
ShowMessage	text (Type: String) title (Type: String) buttonText (Type: string[])	Opens the message prompt with text, title and button options	

Table Component

Class Name: TableHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
Name	String	Get the Name of the Table Component	

Method

Name	Parameter	Description	Pivotal Agent Step
TickleTable	tableName (Type: String)	Tickle Table Data	Tickle Table
Average	tableName (Type: String) columnName (Type: String) filterColumnName	Calculate Average of columnName for filter column value	Static Function Average

	(Type: String) filterColumnValue (Type: Object)		
Count	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate count of rows for filter column value	Static Function Count
Max	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate max value of column name field for filter column value	Static Function Max
Min	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate min of column name field for filter column value	Static Function Min
SQLFind	tableName (Type: String) filterColumnName (Type: String) filterColumnValue	Returns the Record Id for first column filter value find	Static Function SQLFind

	(Type: Object)		
SQLIndex	tableName (Type: String) columnName (Type: String) recorded (Type: Id)	Returns the column name field value for the record	Static Function SQLIndex
SQLSum	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate sum of column name field for filter column value	Static Function SQLSum
UserInGroup	user (Type: object) group (Type: object)	Return Boolean whether a user belongs to a group	Static Function UserInGroup

Variable Component

Class Name: VariableHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
Name	String	Get the Name of the Variable Component	
AttachmentName	String	Attachment name in case variable is of type attachment	
Type	VariableType	Type of Attachment	Type
Value	Object	Variable object value	Value

Move Component

Class Name: MoveHelper
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client

Properties

Name	Type	Description	Pivotal Agent Property
Name	String	Get the Name of the Move Component	

Method

Name	Parameter	Description	Pivotal Agent Step
AddData	6 overloaded method to add data to other object Component property to other Component property Component property to variable Variable to Component property Variable to Variable Static object value to variable Static Object value to component property	Add Data	Add
CopyData	6 overloaded method to copy data to other object Component property to other Component	Copy Data	Move

	property Component property to variable Variable to Component property Variable to Variable Static object value to variable Static Object value to component property		
GetSourceComponentValue	MoveType (Type: MoveType) sourceObject (Type: object) sourceProp (Type: String)	Evaluate component property/field value	

Static Function

Pivotal Agents offers a set of static function that can be used in the expressions to move data from one component to other component. These are the categories of the function available in Pivotal agent

- Information
- Mathematical
- Query
- Statistical
- Date and Time
- Text

Information

Class Name: FInformation
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client.Function

Function

Name	Parameter	Description	Pivotal Agent
CurrentUserId	None	Returns current user id	CurrentUserId()
CurrentUserName	None	Returns current user name	CurrentUserName( )
UserInGroup	user (Type: object) group (Type: object)	Returns True if the user belongs to group else false	UserInGroup(user, group)

Mathematical

Class Name: FMathematical
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client.Function

Function

Name	Parameter	Description	Pivotal Agent
Abs	number (Type: object)	Returns absolute value of number	Abs(number)
Truncate	number (Type: object)	Truncate the number	Truncate(number)
Round	number (Type: object)	Round the number to significant value	Round(number)
RoundTo	number (Type: object) position (Type: object)	Rounds the number to the specified number of decimal place	RoundTo(number, digits)
Remainder	number1 (Type: object) number2 (Type: object)	Returns the remainder when x is divided by y	Remainder(x,y)

Query

Class Name: FQuery
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client.Function

Function

Name	Parameter	Description	Pivotal Agent
SQLfind	tableName (Type: string) filterColumnName (Type: string) filterColumnValue (Type: object)	Returns primary key id for the filter column having filter column value	SQLFind(col, value)
SQLIndex	tableName (Type: string) filterColumnName (Type: string) recordId (Type: Id)	Returns column name value for index row	SQLIndex(col, ID)
CurrentUserRef	tableName (Type: String) columnName (Type: String)	Returns the record id of the record whose columnName contains the current user id	CurrentUserRef(col)

Statistical

Class Name: FStatistical
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client.Function

Function

Name	Parameter	Description	Pivotal Agent
SQLSum	tableName (Type: String) columnName (Type: String) filterColumnName	Calculate sum of columnName (non-null value)for filter column value	SQLSum(col, qf, qid)

	(Type: String) filterColumnValue (Type: Object)		
Count	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate count of rows of columnName (non-null value) for filter column value	Count(col, qf, qid)
Average	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate Average of columnName (non-null value)for filter column value	Average(col, qf, qid)
Max	tableName (Type: String) columnName (Type: String) filterColumnName (Type: String) filterColumnValue (Type: Object)	Calculate max value of column name field (non-null value) for filter column value	Max(col, qf, qid)
Min	tableName (Type: String) columnName (Type: String) filterColumnName	Calculate min value of column name (non-null value) field for filter column value	Min(col, qf, qid)

	(Type: String) filterColumnValue (Type: Object)		

DateTime

Class Name: ExDateTime
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client.Function

Function

Name	Parameter	Description	Pivotal Agent
Today	None	Returns today's date formatted as per windows regional setting	Today()
Now	None	Returns local datetime as per windows setting	Now()
NowGMT	None	Return the current datetime as per GMT	NowGMT()
Date	month (Type: int) day (Type: int) year (Type: int)	Construct a datetime field with month, day and year value	Date(month, day, year)
Day	date (Type: DateTime)	Return day component of datetime field	Day(date)
Month	date (Type: DateTime)	Return month component of datetime field	Month(date)
Year	date (Type: DateTime)	Return year component of datetime field	Year(date)
DayOfWeek	date (Type: DateTime)	Return day of week component	DayOfWeek(date)

		of datetime field	
DayOfYear	date (Type: DateTime)	Return day of the year of datetime field	DayOfYear(date)
Time	hour (Type: int) minute (Type: int) second (Type: int)	Construct a datetime field with hour, minute and second component	Time(hour, minute, second)
Hour	date (Type: DateTime)	Return hour component of datetime field	Hour(time)
Minute	date (Type: DateTime)	Return minute component of datetime field	Minute(time)
Second	date (Type: DateTime)	Return second component of datetime field	Second(Time)
WorkingDays	fromDate (Type: DateTime) toDate (Type: DateTime)	Return the number of working day between from and to date	WorkingDays(from_date, to_date)
AddWorkingDays	date (Type: DateTime) days (Type: int)	Returns working day from the day adding number of days	AddWorkingDays(date, num_days)
ShortDateFormat	date (Type: DateTime)	Returns date to short date format	ShortDateFormat(date)
LongDateFormat	date (Type: DateTime)	Returns date to long date format	LongDateFormat(date)

Text

Class Name: FText
Namespace: CdcSoftware.Pivotal.Applications.AgentRunTime.Client.Function

Function

Name	Parameter	Description	Pivotal Agent
Length	str (Type: string)	Returns length of the string	Length(string)
Left	str (Type: string) number (Type: int)	Returns left number string from input string	Left(string, number)
Right	str (Type: string) number (Type: int)	Returns right number string from input string	Right(string, number)
Mid	str (Type: string) Start (Type: int) length (Type: int)	String string with start to length number of input string	Mid(string, start, number)
Trim	str (Type: string)	Trims leading and trailing blanks from string	Trim(string)
Value	str (Type: string)	Parse string to decimal value	Value(string)
Search	findStr (Type: string) inStr (Type: string)	Returns the starting position of searched string in the string	Search(findstring, instring)
Lower	str (Type: string)	Convert string to lower case	Lower(string)
Upper	str (Type: string)	Convert string to upper case	Upper(string)
Replace	str (Type: string) oldText (Type: string)	Replace first instances of oldtext with nexttext in the	Repalce(string, oldtext, newtext)

	newText (Type: string)	string	
ReplaceAll	str (Type: string) oldText (Type: string) newText (Type: string)	Replace all instances of oldtext with nexttext in the string	ReplaceAll(string, oldtext, newtext)
ProperCase	str (Type: string)	Capitalize the first character of each word	ProperCase(text)
Format	format (Type: String) str (Type: String)	Format the string as per specification of the format	Format(specification,text)
Soundex	str (Type: string)	Returns the soundex code for the text	Soundex(text)
IsAlphabetic	str (Type: String) start (type: int) length (Type:int)	Returns boolean flag to indicate whether string of length with starting number is alphabetic	IsAlphabetic(string, start, number)
IsNumeric	str (Type: String) start (type: int) length (Type:int)	Returns boolean flag to indicate whether string of length with starting number is numeric	IsNumeric(string, start, number)
IsAlphaNumeric	str (Type: String) start (type: int) length (Type:int)	Returns boolean flag to indicate whether string of length with starting number is alphanumeric	IsAlphaNumeric(string, start, number)
LocalizeString	name (Type: string) groupName (Type:	Returns the translated string for language	LocalizeString(string name, string group name,

	string) defaultString (Type: string)	string and group	default string)