



AIM Integration Framework

Setup and User Guide

Revised: 10.7.2014

© 2014 Vertafore, Inc. and its subsidiaries. All Rights Reserved. Vertafore, the Vertafore design, AfW, AMS360, BenefitPoint, ClientConnect, Compliance Express, Engage, FinancePro, FSC Rater, ImageRight, ImageRight FastTrack, Prevail Network, Producer Express, Producer Plus, ReferenceConnect, Sagitta, SilverPlume, Sircon, Sircon for States, TransactNOW, and WorkSmart are all registered trademarks of Vertafore, Inc. or its subsidiaries.

Contents

Introduction	1
Prerequisites	1
Setup AIM Integration Framework.....	1
Prepare Aim Integration Framework.....	1
Enable AIM Integration Framework	1
API	2
Command Types.....	3
XML Commands.....	3
Custom Action.....	3
Sample XML Input.....	4
Search AIM Objects.....	5
Sample XML Input.....	5
Add AIM Object.....	5
Database Tables	6
CIS.dbo.AimIntegration.....	6
CIS.dbo.AimIntegrationHistory	6
CIS.dbo.AimIntegrationMap	7
AIM Custom Actions	8
Use Custom Actions.....	8
Database Table	8
Item Display	8
Interaction.....	8
Component Interface.....	8

Introduction

AIM Integration Framework is a way for AIM to integrate with external systems for the purpose of passing basic data. The integration framework can also be used to create custom menu commands.



Data from external systems does not come back into AIM. That is, the transfer of data is a one-way transfer from AIM into an external system.

Prerequisites

Prior to configuring AIM Integration Framework, you must obtain the following components required for installation.

- Obtain Microsoft .NET version 4 from Microsoft
- Contact Vertafore and obtain the following:
 - AIMIntegrationInstaller.msi
 - Database scripts for AIM Integration Framework

Setup AIM Integration Framework

In this section we discuss the 4 main steps for setting up AIM Integration Framework. These steps are discussed at a high level. It is expected that you are capable of performing the steps as outlined.

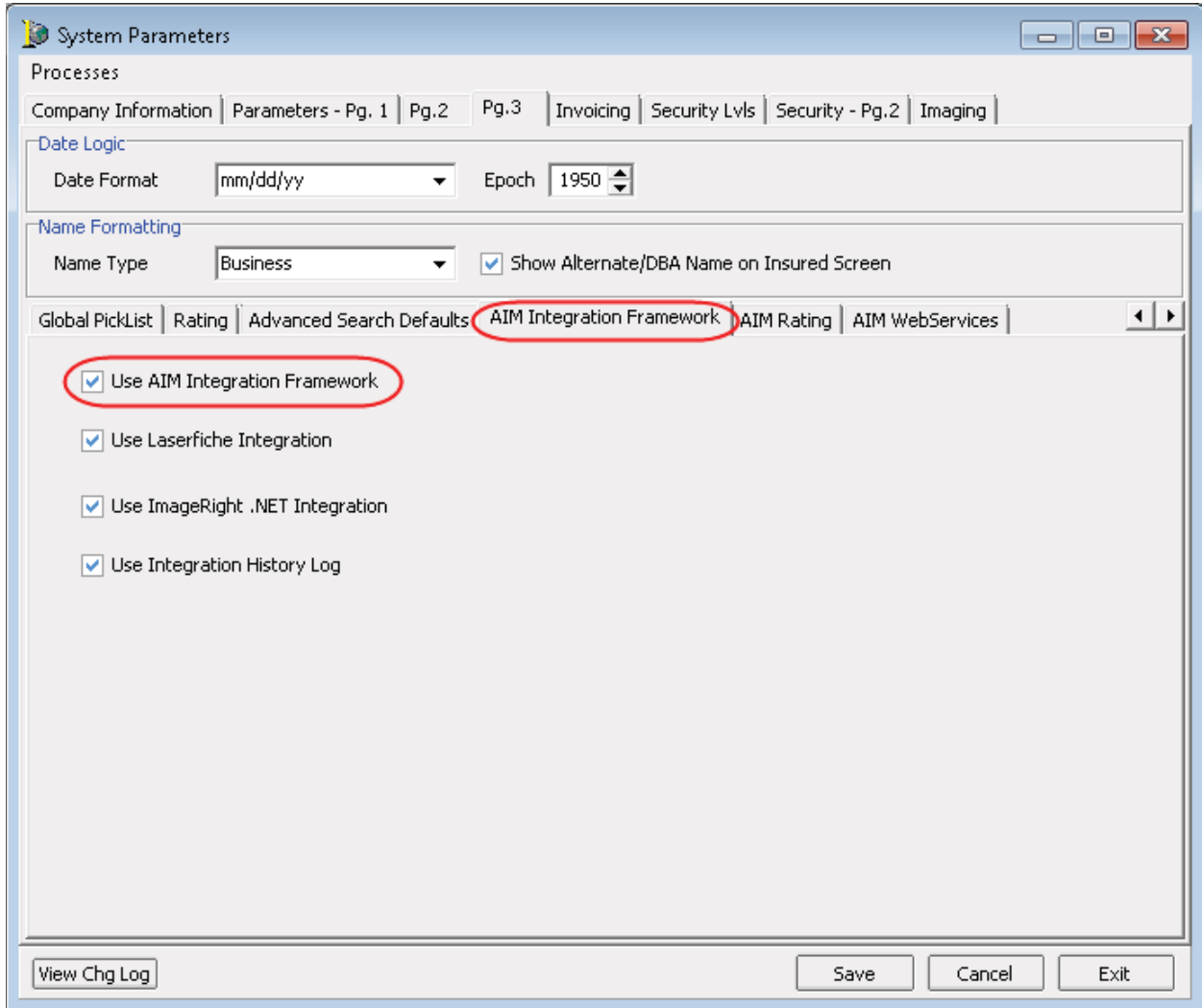
Prepare Aim Integration Framework

1. Install the .NET Framework version 4.
Prior to proceeding, you will need to install Microsoft .NET Framework Version 4 on the server or workstation.
2. Run the provided installation file named AimIntegrationInstaller.msi.
3. Run the database change scripts to create the tables AimIntegration, AimIntegrationMap and AimIntegrationHistory.
4. Configure the commands in the AIMIntegration table if needed. The database change scripts input entries into the database for the given commands however; you may wish to modify the entries as needed.

Enable AIM Integration Framework

1. Open Data Maintenance Utility (DMU).

2. On the **Setup** menu, click **System Parameters**.
3. On the **Pg. 3** tab, click the **AIM Integration Framework** tab.
4. Select the **Use AIM Integration Framework** option, shown in Figure 1 below.
5. If using Laserfiche integration, both options need to be selected.



The screenshot shows the 'System Parameters' dialog box with the 'Pg. 3' tab selected. The 'AIM Integration Framework' sub-tab is active. The 'Date Logic' section shows 'Date Format' as 'mm/dd/yy' and 'Epoch' as '1950'. The 'Name Formatting' section shows 'Name Type' as 'Business' and 'Show Alternate/DBA Name on Insured Screen' checked. The 'AIM Integration Framework' sub-tab is selected, and the 'Use AIM Integration Framework' checkbox is checked and circled in red. Other checked options include 'Use Laserfiche Integration', 'Use ImageRight .NET Integration', and 'Use Integration History Log'. The 'Global PickList', 'Rating', 'Advanced Search Defaults', 'AIM Rating', and 'AIM WebServices' tabs are also visible.

Figure 1: AIM Integration Framework tab

API

Below are the functions for the AIM Integration Framework.

```
void SetSQLConnectionString(string serverName, string userName, string password);
string GetValueForOption(string commandName, string optionName);
void SetOptionForCommand(string commandName, string optionToSet, string valueToSet);
string RunCommand(string commandName, string commandType, string userName, string key);
string Start();
```

Command Types

Below we describe the command types used in AIM Integration Framework.

- **Standard Commands** – Runs a command, such as executing an exe file or saving an xml file to disk. The result is added to the History table if the option in the DMU is checked.
- **XML Commands** – Takes in XML input, runs the command, and returns XML output. The XML input should be put into the Key parameter. Inside the XML in the XMLCOMMANDNAME property is the command to be run. The resulting XML is also added to the History table if that is turned on.
- **Event** – A command that is sent from AIM to AIM Integration Framework whenever a specific AIM event occurs. By default, the only event implemented is ONSUBMISSION, which is occurs whenever a submission is created. By itself, the event does nothing, but when tied to an event in the AimIntegration table, when the event is fired whatever commands that are tied to the event are fired as well.

XML Commands

In the table below, we discuss the XML commands used in AIM Intergration Framework.

Custom Action

This calls a 3rd party COM object. The COM object needs to have the following functions defined:

- Init
- Display
- DeInit.

These functions will be called in that order.

Setting Name	Required (Y/N)	Accepted Values
XMLCOMMANDNAME	Y	Custom Action

Setting Name	Required (Y/N)	Accepted Values
PROGID	Y	A string containing the PROGID of the COM object to instantiate.
SUBMISSIONID	N	The submission ID of the submission which was currently open before making the custom action call. This is blank if none were open.
QUOTEID	N	The quote ID that was currently open before making the custom action call. This is blank if none were open.
VERSION	N	The version of the quote that was currently open before making the custom action call. This is blank if none were open.
POLICYID	N	The ID of the policy that was currently open before making the custom action call. This is blank if none were open.
NEXTACTION	N	This is blank in the input and is filled out by the COM object. This is the next action to perform by AIM once the call to custom action has been completed.

Sample XML Input

```

<?xml version="1.0" encoding="utf-16"?>
<XMLCOMMAND>
<NAME>CUSTOM ACTION SAMPLE</NAME>
<OPTIONS>
  <COMMAND>XML COMMAND</COMMAND>
  <XMLCOMMANDNAME>CUSTOM ACTION</XMLCOMMANDNAME>
  <QUOTEID>060003</QUOTEID>
  <VERSION>A</VERSION>
  <PROGID>CustomActionSample.CustomActionSample</PROGID>
</OPTIONS>
</XMLCOMMAND>

```

Search AIM Objects

Searches a table and returns all of the matching records. This function does not return every field in the table, only those specified in the FIELDLIST property.

Setting Name	Required (Y/N)	Accepted Values
XMLCOMMANDNAME	Y	Search AIM objects
SEARCHTABLE	Y	The table that will be searched, for example: Quote, Suspense, etc.
SEARCHFIELD	Y	The field that will be searched. Currently only text fields can be used for searches. For example: QuoteID
SEARCHVALUE	Y	The text value to be searched for in the search field. For example: 060003
FIELDLIST	Y	This is the listing of fields that will be returned for all matching records. Currently they have to be text fields.

Sample XML Input

```
?xml version="1.0" encoding="utf-16"?>
<XMLCOMMAND>
  <NAME>SEARCH AIM OBJECTS SAMPLE</NAME>
  <OPTIONS>
    <COMMAND>XML COMMAND</COMMAND>
    <XMLCOMMANDNAME>SEARCH AIM OBJECTS</XMLCOMMANDNAME>
    <SEARCHTABLE>Quote</SEARCHTABLE>
    <SEARCHFIELD>ProducerID</SEARCHFIELD>
    <SEARCHVALUE>AGT009</SEARCHVALUE>
    <FIELDLIST>QuoteID,NamedInsured,TeamID,DivisionID</FIELDLIST>
  </OPTIONS>
</XMLCOMMAND>
```

Add AIM Object

This functionality is not yet fully implemented.

Setting Name	Required (Y/N)	Accepted Values
XMLCOMMANDNAME	Y	Add AIM object

Database Tables

Below we describe the database tables related to the AIM Integration Framework.

CIS.dbo.AimIntegration

This table controls the integration commands and their options which are loaded during runtime. This table is modified to change the options for commands.

Column Name	Description	Example Value
AimIntegrationID	Numeric primary key which is auto generated.	1
UserSpecificCommandName	The user given name of the command. Make sure that this is a unique name for each command. All of the setting options for a specific command must have the same name.	LFRUNEXE
Option	This is the name of the option which will be set.	EXEPATH
Value	This is the value of the option which will be set. So for this example we are giving the option EXEPATH the value of C:\Test\Test.exe .	C:\Test\Test.exe

CIS.dbo.AimIntegrationHistory

This table contains the results and history information for all commands that are ran. This table is automatically updated by the application.

Column Name	Description	Example Value
AimIntegrationHistoryID	Numeric primary key which is auto generated.	1
UserSpecificCommandName	The user given name of the command.	LFRUNEXE
Result	This is the result of the command. SUCCESS means that the command ran without any errors. If there was an error than this is where it would display the error.	SUCCESS

Column Name	Description	Example Value
DateRan	This is the Date/Time that the command ran.	2010-05-18 14:32:11
User	This is the user that ran the command.	LSMITH
OptionsUsed	This is a string containing all of the options for the command which were run. This is intended for troubleshooting purposes.	([COMMAND = RUNEXE][EXEPATH = C:\TEST\TEST.exe] [PARAMTYPE = LASERFICHE][USE RNAME = LSMITH][KEY = 00001][PARAMS = LSMITH 0001])

CIS.dbo.AimIntegrationMap

This table holds mapping values used by the integration. For example if using ImageRight integration this can hold the mappings from AIM users to their appropriate ImageRight drawers.

Column Name	Description	Example Value
AimIntegrationID	Numeric primary key which is auto generated.	1
MappingType	The type of mapping, this must be a certain value depending on the integration.	IRAIMMapping
SourceTable	This is the source table or object.	AIM
SourceColumn	This is the source column or property.	CIST
DestinationObject	This is the destination object or table.	Drawer
DestinationColumn	This is the destination column or property.	CUST

AIM Custom Actions

AIM Custom Actions adds custom menu items to AIM that can be selected by the user and called by AIM. The main purpose is so that AIM can integrate with 3rd party/client components. These 3rd party components need to conform to the technical specifications outlined in this document so that they can be called from AIM. These custom actions are implemented by using the AIM Integration Framework.

AIM passes the current submission and quote version to the component and the component then displays a form in a modal state until it is finished. The component then returns XML to AIM that describes what submission to change to (if any) and what button or menu item to select (if any). (Edit Submission, Edit Insured, etc.) This component is a standard COM component called by AIM using its ProgID.

Use Custom Actions

1. In AIM click **Custom Menu** located between **Detail** and **Help**.
The custom menu items that you setup appear below. If there are no custom actions set up in the database then this menu is hidden.
2. Click the desired custom menu option and the custom action associated with that menu item will be called.

Database Table

A database table is created to hold the menu item text, the ProgID of the component, and an order number to indicate what order the items should be displayed in. At this time, there is no user interface to maintain this table.

Item Display

The menu items should be displayed using the order number from the table to indicate display order. These new menu items should be placed between the **Detail** and **Window** menu commands with header text of **Custom**. Once a user clicks on the **Custom**, the menu items in the table will be displayed below just as any other menu item.

Interaction

When you click an individual item, the component becomes instantiated if it has not already been. The **Init** method is called followed by the **Display** method. The **Display** method returns an XML string to describe what submission to change too and what action to display next. Upon shutdown of AIM, each of the components will have their **Delnit** method called.

Component Interface

Void Init(void) – This method is used by the component to signal a reset of any internal resources.

Void DeInit(void) – This method is used by the component to signal a shutdown of AIM to release any internal resources.

String Display(String xml) – The display method receives an XML stream as an argument that tells it what the current context is. It then displays a modal form to hold focus until finished executing. Then it returns an XML stream that tells AIM what submission to change too. If the argument is returned blank, then no change is necessary. The XML stream also contains what action to execute next. AIM executes certain menu items after changing the submission context. These actions include:

- **Edit Submission**
- **Edit Insured**
- **Quote** button
- **Bind** button
- **Policy Endorse** menu command

If the action item is blank, then no action is performed.