Implementing OpenSpan Agile Desktop

for OpenSpan 7.0 and higher

20 January 2015

This manual is designed for OpenSpan solution developers. It focuses on setting up, using, and deploying OpenSpan Agile Desktop. Use this table to quickly find the information you need in this manual:

Public

©Copyright 2012 - 2016 OpenSpan, Inc. All Rights Reserved.

No part of this publication may be reproduced or distributed in any form or by any means, electronic or otherwise, now known or hereafter developed, including, but not limited to, the Internet, without explicit prior written consent from OpenSpan, Inc. Requests for permission to reproduce or distribute to individuals not employed by OpenSpan any part of, or all of, this publication should be mailed to:

OpenSpan, Inc.
Suite 200
11175 Cicero Drive
Alpharetta, Georgia 30022

Phone (International) +1 (678) 527-5400
Phone (US and Canada) (877) 733-1136
Email: sales@openspan.com

The following are trademarks or registered trademarks of OpenSpan Inc., a Delaware Corporation:

OpenSpan®
Where people and technology meet®
A better way to manage®
A better way to work®
OpenSpan Workforce Intelligence™

Microsoft®, Visual Studio®, MSDN®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Public

Contents

Public

Chapter 1: Implementing Agile Desktop

Welcome to OpenSpan Agile Desktop. This tool, built for developers who use OpenSpan Studio version 7.0 or higher to create solutions for OpenSpan customers, provides you with a framework for presenting the solutions you design to the end user.

This framework ensures a user-friendly and attractive design and one that brands your work as an OpenSpan solution.

This guide includes these topics:

Terminology

accent color

Agile Desktop lets you specify a color to be used in the interface. Typically this color is one of the end user's corporate colors. You specify this color in the accentColor option in the RuntimeConfig.xml file. Based on the color you choose, different shades of that color are used for different user interface elements, as shown in this table:

Control	Percentage of Accent Color
Closed sidebar	100
Social dots	100
Shortcut button (normal) (pressed border)	100
Command button (normal) (pressed border)	100
No Historical Notes (heading)	100
Identification field in 360 view	100
Shortcut button (hover) (pressed)	105
Command button (hover) (pressed)	105
Interaction drop-down (current interaction)	100
Interaction drop-down (current interaction - hover)	105
Interaction drop-down (not current interaction)	60
Interaction drop-down (not current interaction - hover)	80

activities

The various work items that constitute the interaction are called *activities*. Changing an address is an example of an activity. Activities are queued and only one activity can run at a time. Because only one activity happens at a time, you do not have to worry about the processing of one activity affecting the another activity by changing the state of an application.

Activities are defined in the Activities section of the interaction.xml file. You can define as many activities as you need in this section and every defined activity is represented by its own Activity component.

When a method starts an activity in one deployed package, it can fire an event within the same deployed package or in a different deployed package. Activities are queued and run in sequence within the Interaction Framework.

Context

This refers to a section in the interaction.xml file that defines the properties Interaction Manager will track and make available to automations. Make sure all of the properties your automations will use are defined in this section.

CSS

Cascading Style Sheets (CSS) is a style sheet language used to define the formatting of a document written in a markup language. While CSS is generally used to control the style of web pages, OpenSpan uses CSS, along with HTML and JavaScript, to create the user interface for Agile Desktop.

end user

This term refers to the person using Agile Desktop. This person could work in a call center or the back office of a financial concern, for instance. This person may communicate with his or her company's clients, called customers in this document.

extended view

This view appears when you click the arrow in the primary view. This view provides additional information.

customer

The person the end user interacts with. For instance, an end user could be a customer service representative who is using Agile Desktop. The person the end user talks to would be the customer.

Interaction Manager

This component connects all customer interactions through the configuration of its properties, events and methods. The Interaction Manager maintains a direct link to the contents of the interaction.xml file and stores the context values defined for each interaction.

These context values appear as component properties. Only one Interaction Manager component can be referenced per project in a deployment package, and it enables the connectivity of the framework's activities and tasks.

interactions

An interaction is a session with a customer. The session could be a phone call or a chat, for instance. An interaction is sometimes called a transaction. Once an interaction is started, activities can be started and stopped within that interaction.

A person can have multiple interactions going on at a time. For instance, an agent could put one customer on hold, while resolving another call.

interaction.xml file

This file defines the contexts, activities, and plug-ins that comprise the Agile Desktop implementation you are designing. This file serves as a binding contract that allows for communicating information and events across independently running deployed packages. Typically, there will only be one, but you can define multiple interaction XML configuration files.

Note You can find a sample interaction file in the AgileDesktop/Samples/Config folder. For information on the options in this configuration file see the Using the Interaction.xml File topic in Help.

package

A package is a deployable solution ready to be used by Agile Desktop. A deployment package is comprised of two files, with these extensions: .openspan and .manifest. The .openspan file contains the specific adapter, translators, and custom components that make up the solution. The .manifest file contains a list of the contents of the .openspan file, along with project version information.

plug-in

In the context of this document, a plug-in is a component that adds a feature to Agile Desktop, such as the ability to take and save notes about the interaction. OpenSpan Agile Desktop comes with three out-of-the-box plug-ins: 360 View, Notes, and Shortcuts. These plug-ins are built with HTML with Java scripting.

project

OpenSpan Studio bundles functionality in project items. OpenSpan project items let you integrate applications, monitor events, and automate tasks. Adapters, automations, and global containers are stored as .os and .xml files in the solution and project folders.

solution

Within OpenSpan Studio, solutions and projects are containers for all of the items required to build and run the project. Solutions can include multiple projects. Once you deploy a solution to OpenSpan Management Console (OMC), it is known as a *package*.

styling

This refers to customizing the way values are shown in Agile Desktop to conditionally emphasize those values. For instance, you could display an alert if a customer's balance fell below a given amount. You can also rank items.

primary view

The initial view an end user sees when Agile Desktop starts.

UAD

Unified Agent Desktop. A UAD is an application which attempts to bridge the multiple applications an end user works with. The 360 View plug-in can provide the same types of information and integration without the heavy development commitment required when creating a UAD plus, with OpenSpan Studio, it is much easier to make changes which reflect the end user's ever changing environment.

view

An Agile Desktop plug-in can have two views: primary and extended. The extended view is optional and lets you present more information to the end user if the user clicks the icon to expand the primary view.

Overview

Agile Desktop provides a framework for integrating, managing, and presenting the OpenSpan solutions you create. Agile Desktop includes several predefined solutions that let you track notes, present shortcuts, and provide a panoramic view of the current customer. Agile Desktop provides a set of user interface elements so you can design an easy-to-use, standardized, and configurable end user experience for those using OpenSpan projects.

Agile Desktop hosts and runs OpenSpan projects. Agile Desktop shares data and orchestrates actions across projects without introducing direct project dependencies.

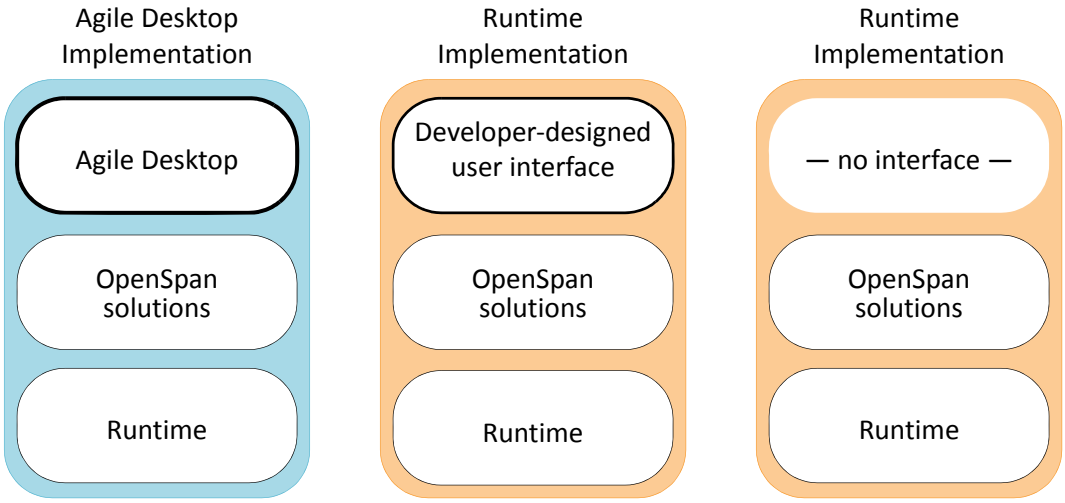
In addition to simplifying the development of new features by building configuration, analytics, and other capabilities into a standard framework, Agile Desktop provides a...

- Modern, easy-to-use look and feel.
- Recognizable, standardized OpenSpan identity on the desktop.
- Way to coordinate projects without creating direct dependencies.
- Framework that encourages project standardization in Services.

Agile Desktop hosts and displays HTML/JavaScript user interface plug-ins. Its user interface is a .NET control rendered in HTML using C#. The core UI is single-threaded and it too is comprised of HTML/JavaScript plug-ins.

What's the difference between Agile Desktop and Runtime?

Both Agile Desktop and Runtime provide a way to make the OpenSpan solutions you develop available to users. Both share much in common, as this illustration shows:



The difference is in the user interface.

With Runtime, the user interface is optional and is something an OpenSpan developer must create. With Agile Desktop, much of the user interface is created for you by OpenSpan. You only have to customize Agile Desktop to meet the needs of your implementation.

Agile Desktop's presentation system lets you quickly define and display data from your OpenSpan solutions in a collapsible sidebar. This sidebar is rendered in HTML and styled using a CSS file. Thus, one of the first things Agile Desktop does when it starts is read the HTML and CSS presentation files. If it does not find these files, located in the AgileDesktop folder, you will receive an error message. For more information see "Deploying Agile Desktop to End Users" on page 50.

Note OpenSpan Agile Desktop and Runtime are installed in the same program folder — \OpenSpan Runtime Enterprise. While there are separate executable files, both Agile Desktop and Runtime use the same supporting libraries and configuration files to run OpenSpan solutions.

Who should use it?

We recommend that all OpenSpan customers use Agile Desktop. All OpenSpan Services developers will use Agile Desktop.

OpenSpan is committed to providing a modern, standardized, and user-friendly means of presenting OpenSpan solutions to end users and also branding these tools with the OpenSpan name to improve our standing in the marketplace.

How Does It Work?

The Interaction Framework lets Agile Desktop load multiple projects that interact with each other using a decoupled publish-subscribe system. Each project can interact with the interaction component to initiate new interactions and activities or it can be notified of new interactions or activities.

Agile Desktop supports multiple concurrent interactions or serial single interactions. For instance, in a contact center, the interaction will typically be the call itself, whereas in the back office the interaction will be the current transaction. Note that the Interaction-Activity framework also forms the foundation for analytics projects and will be available outside the Agile Desktop solution offering.

Defining Contexts

Interactions expose a context, a shared set of data that can be updated throughout the call or transaction. Any automation can update the interaction context at any time or subscribe to context updated events. Whenever an activity, application usage, or application detail event is sent to the server, it will automatically capture and send the associated interaction context.

Initiating Activities

Interactions also allow automations to initiate activities or be notified of activities. This lets automation developers participate in shared activities, such as search, without the need for a single, coordinating parent project. Activities can include a shared set of data that all automations can read or update. Whenever an activity is executed in the interaction-activity framework, an activity event is raised. The activity event includes the current context data. All activities are subsequently available to business analysts to attach to notes, offers, and so on.

Configuring the Interaction

The interaction.xml file defines the context, activities, and associated plug-ins. These definitions are used at design time to populate a set of methods to update the context and to execute activities. OpenSpan Studio creates these methods for your convenience. The definitions are retrieved at runtime to populate notes, shortcuts, and so on.

Keep in mind that the interaction-activity framework does not enforce the accuracy of activity and context definitions. If an activity is raised that is not in the definition, it is ignored by all participants that do not recognize it.

These definitions are also used by Agile Desktop plug-ins to render shortcuts, data, and so on.

Synchronizing the configuration

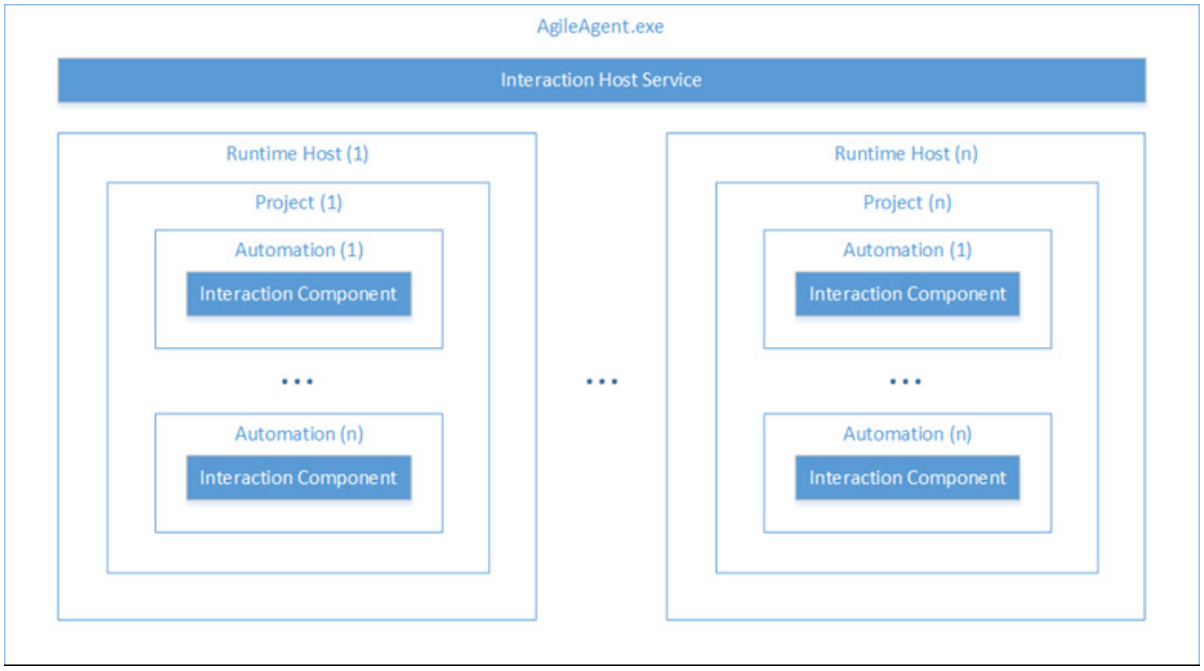
The interaction.xml configuration file is versioned and stored on the OpenSpan Management Console (OMC) so it can be shared by automation authors and business analysts. When using the interaction component in a solution, the default interaction.xml file is added to solution. You can either edit it or browse to select a different one.

Understanding the Interaction Host

The interaction host is the service responsible for coordinating interactions and managing packages. At startup, the interaction host performs these tasks...

- Initializes the user interface
- Contacts the configuration server
- Downloads the interaction configuration and all assigned packages
- Initializes the set of packages assigned to the end user

The interaction host also enables communication between the interaction components hosted in each project. This illustration shows how it works:



Executing Activities

The interaction host coordinates all requests for new interactions and activities. The interaction host is provided to all hosted projects as a service that the interaction host can interact with and receive events.

To avoid scenarios where multiple interactions or activities are requested simultaneously, the interaction host queues all interaction initialization and activity execution. Automation authors must notify the interaction host using an interaction component when starting and ending execution to enable this functionality.

Automation authors must also check the activity state at appropriate break points to see if long-running activities have been cancelled. The interaction host sends the appropriate events to the server whenever...

- An interaction is initialized
- The interaction context is updated
- An activity is executed or cancelled

Note that it may be possible to enhance automations so completion and cancellation are handled automatically.

Using the Interaction Component

The interaction component is the primary interface between automation authors and the interaction host. Using the interaction component automation authors can initiate interactions, update or read the interaction context, or raise activity events.

Using Context Properties and Methods

The interaction component generates properties so automation authors can get and set context values. For example, if the context is configured with these properties: FirstName, LastName, and FullName, the interaction component will contain these dynamically generated properties: FirstName, LastName, and FullName. Each of these generated properties invokes the methods (also exposed) defined here:

```
int GetFactValue(string key);
void SetFactValue(string key, int value);
string GetDimensionValue(string key);
string SetDimensionValue(string key, string value);
string GetAttributeValue(string key);
string SetAttributeValue(string key, string value);
```

Using Activity Methods

The interaction component provides methods so automation authors can invoke activities or notify the interaction host when activities are started, completed, or cancelled. For example, if the interaction configuration includes a Search activity, these methods and events appear on the interaction component:

- InvokeSearch
- SearchInvoked
- CancelSearch
- SearchCancelled
- NotifySearchStart
- NotifySearchComplete

Each of these generated methods invoke the methods and events (also exposed) defined here:

```
int InvokeActivity(string name, params object[] args);
event EventHandler<ActivityArgs> ActivityInvoked;
void NotifyActivityStart(int id);
void NotifyActivityCompleted(int id);
event EventHandler<ActivityArgs> ActivityCompleted;
void CancelActivity(string id);
event EventHandler<ActivityArgs> ActivityCancelled;
```

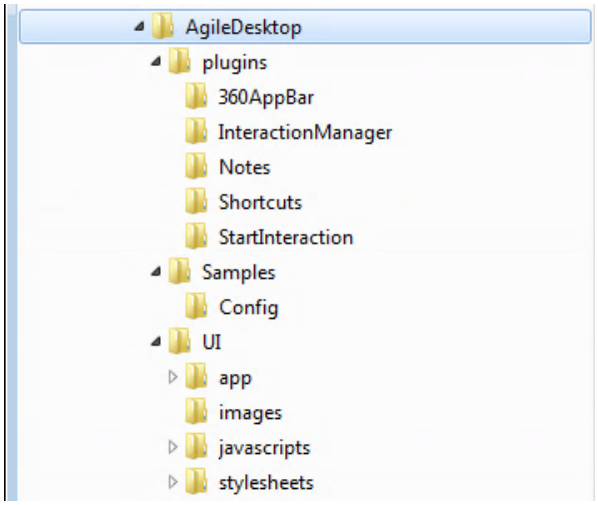
Using Interaction Methods

The interaction component provides these methods and events so automation authors can initialize or close interactions or be notified when interactions are initialized or closed.

```
void InitializeInteraction(string key);
event EventHandler<InteractionArgs> InteractionInitializeInvoked;
void NotifyInteractionInitializeStarted(string key);
void NofityInteractionInitializeCompleted(string key);
event EventHandler<InteractionArgs> InteractionInitializeCompleted;
void CloseInteraction(string key);
event EventHandler<InteractionArgs> InteractionCloseInvoked;
void NotifyInteractionCloseStarted(string key);
void NofityInteractionCloseCompleted(string key);
event EventHandler<InteractionArgs> InteractionCloseCompleted;
```

How Do I Use It?

The ability to customize the Agile Desktop user interface, along with the three default plug-ins, is built into OpenSpan Studio version 7.0 and higher. Here is the folder structure:



What is required?

The system requirements for running Agile Desktop are minimal. You only need these items:

- Microsoft .NET Framework version 4.0
- OpenSpan Runtime version 7.0 or higher

To use the Agile Desktop user interface, you merely select it as the default application when installing OpenSpan Runtime. See the OpenSpan Runtime Installation Instructions for more information.

How Do I Set It Up?

When you run the OpenSpan Runtime installation wizard, both Agile Desktop and Runtime are installed. Based on your choice in the installation wizard, an icon is created for either Agile Desktop or Runtime. You then use the interaction.xml configuration file to set up Agile Desktop.

Customizing the Interaction.xml File

This file defines the data items that can be shared by the projects in your solution as well as the activities that projects can start.

When an InteractionManager component is added to an OpenSpan Global Container, a copy of the default Interaction.xml file is added to the project folder which you can then edit.

You can name or locate the file anywhere you wish as long as you accurately reference it in the ConfigurationFile property of the InteractionManager component.

You can use a method on the InteractionManager component to start an interaction with a specific name. That name is then used to open the corresponding XML definition file for the interaction.

Note For more information, see the Using the Interaction.xml File topic in OpenSpan Help.

Add this to the Interaction.XML file to separate sections:

```
<Section Name="Products"  Visibility="Both" Divider="True">
```

Within a section, you just add a new Item:

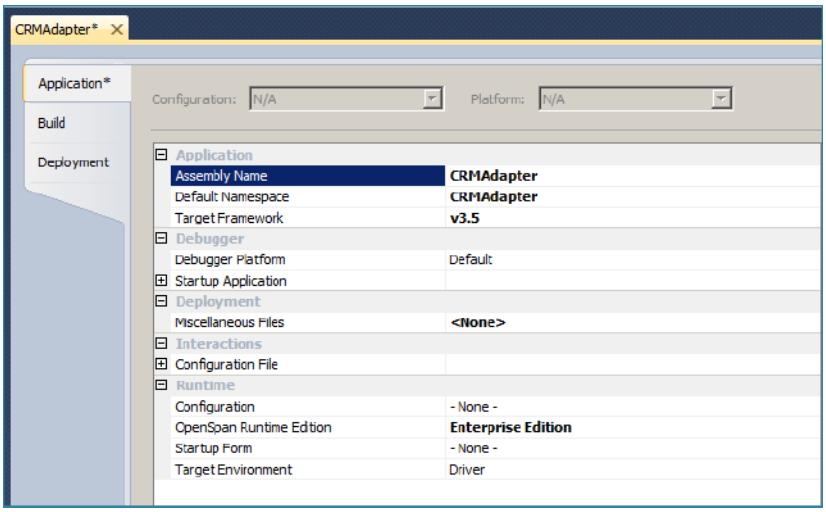
```
<Item Divider="True"/>
```

Note that you must manually add lines in the 360 View section.

Configuring Project Properties

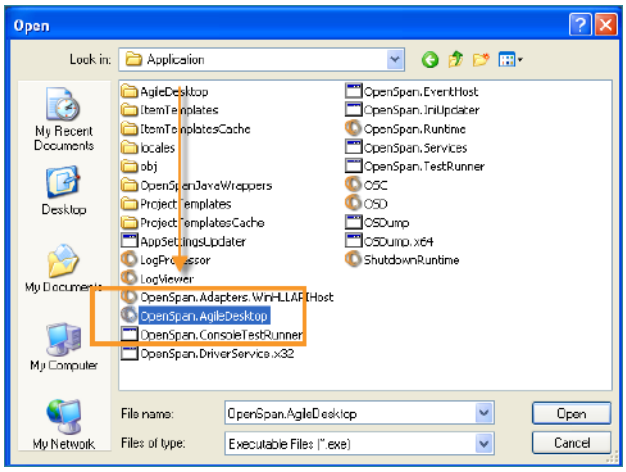
Follow these steps to configure the Agile Desktop project properties.

1. Open the solution you created. The projects you created appear in Solution Explorer.
2. Right-click your project and select Properties. The Project Property pages appear in the Designer.

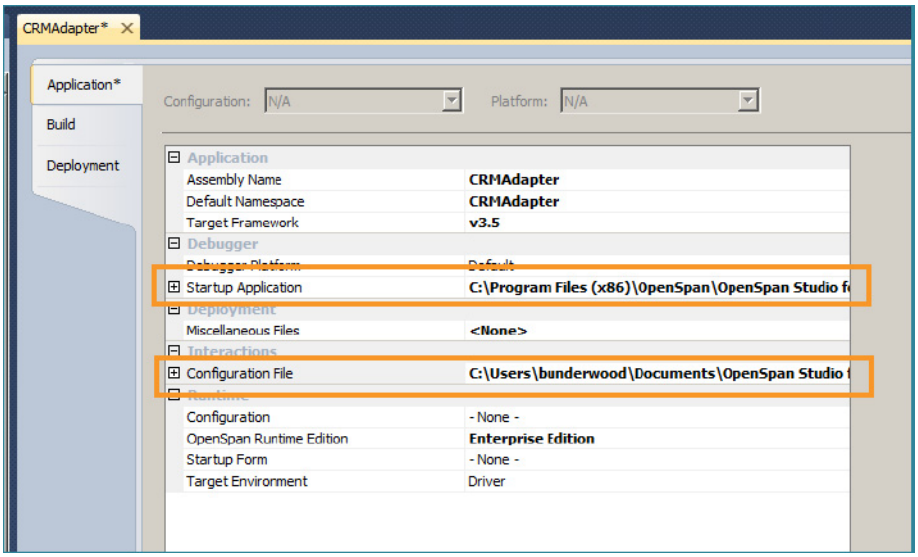


3. Click the Application tab and select the Startup Application field. Next click the (…) ellipse button and set the Startup Application to the OpenSpan.AgileDesktop.exe and click Open.

Note The OpenSpan.AgileDesktop.exe file is located in the Program Files | OpenSpan Studio | Application folder where OpenSpan Studio is installed.



4. Select the Interactions | Configuration File field. Click the (…) ellipse button and set the Configuration File to the Interacton.xml file.



5. Select File | Save All.
6. Repeat steps 2-4 for the rest of your projects to set the StartUp Application and Interaction | Configuration File:
7. Select File | Save All when finished.

Adding a Logo

Follow these steps to modify the logo that displays at the top of the Agile Desktop toolbar.

Note Make a back-up copy of the RuntimeConfig.xml file before you modify this file.

Follow these steps:

1. Locate the RuntimeConfig.xml file in the User | Application Data folder.
2. Highlight and right-click and choose Open With | Notepad.
3. Locate the AgileDesktop section. Here is an example:

```
<AgileDesktop>
        <Url value="AgileDesktop"/>
        <LogoUrl value=""/>
        <UserCanHide value="true"/>
        <UserCanExit value="true"/>
        <TopMost value="true"/>
        <ReadUITimeout value="10000"/>
        <AccentColor value="#DF7A1C"/>
</AgileDesktop>
```

   Here is a summary of the properties in this section:

Property	Description
URL	The location of the Agile Desktop UI and plug-in files. May be a local folder or a URL.
LogoUrl	The location of the logo image file to display in toolbar. May be a local file location or a URL. Spaces are ignored.
UserCanHide	If True, the user can collapse the toolbar to the left side of the desktop.
UserCanExit	If True, the user can exit the toolbar from the gear menu.
TopMost	If True, the toolbar will stay on top of other focused applications.
ReadUITimeout	The number of milliseconds to attempt to read the UI index.dat file before timing out. The default value is 10000
AccentColor	Specify the accent color you want to use in the toolbar user interface. You can choose from any of the HTML color codes. You can enter text values, such as Green or LightBlue, or hex values, such as #00800 or #ADD8E6.

4. Enter as the LogoUrl value the location of the image you want to add. Here is an example:

   ```
   <LogoUrl value="c:\logos\companylogo.png"
   ```

Note For best results, use a logo that is 30 pixels in height.

5. Select File | Save.

Modifying the Accent Color

Follow these steps to change the accent color in the Agile Desktop UI:

Note Make a back-up copy of the RumtimeConfig.xml file before you modify this file.

1. Locate the RuntimeConfig.xml file in the User | Application Data folder.
2. Highlight and right-click and choose Open With | Notepad.
3. Locate the AgileDesktop section. Here is an example:

```
<AgileDesktop>
        <Url value="AgileDesktop"/>
        <LogoUrl value=""/>
        <UserCanHide value="true"/>
        <UserCanExit value="true"/>
        <TopMost value="true"/>
        <ReadUITimeout value="10000"/>
        <AccentColor value="#DF7A1C"/>
</AgileDesktop>
```

Here is a summary of the options in this section:

Property	Description
URL	The location of the Agile Desktop UI and plug-in files. May be a local folder or a URL.
LogoUrl	The location of the logo image file to display in toolbar. May be a local file location or a URL.
UserCanHide	If True, the user can collapse the toolbar to the left side of the desktop.
UserCanExit	If True, the user can exit the toolbar from the gear menu.
TopMost	If True, the toolbar will stay on top of other focused applications.
ReadUITimeout	The number of milliseconds to attempt to read the UI index.dat file before timing out. The default value is 10000
AccentColor	Specify the accent color you want to use in the toolbar user interface. You can choose from any of the HTML color codes. You can enter text values, such as Green or LightBlue, or hex values, such as #00800 or #ADD8E6.

4. Update the value in the AccentColor property. You can choose from any of the HTML color codes. You can enter text values, such as Green or LightBlue, or hex values, such as #00800 or #ADD8E6.

This chart shows common colors:

Color Name	Color Code	Color Name	Color Code
Red	#FF0000	White	#FFFFFF
Cyan	#00FFFF	Silver	#C0C0C0
Blue	#0000FF	Gray or Grey	#808080
DarkBlue	#0000A0	Black	#000000
LightBlue	#ADD8E6	Orange	#FFA500
Purple	#800080	Brown	#A52A2A
Yellow	#FFFF00	Maroon	#800000
Lime	#00FF00	Green	#008000
Magenta	#FF00FF	Olive	#808000

5. Select File | Save.

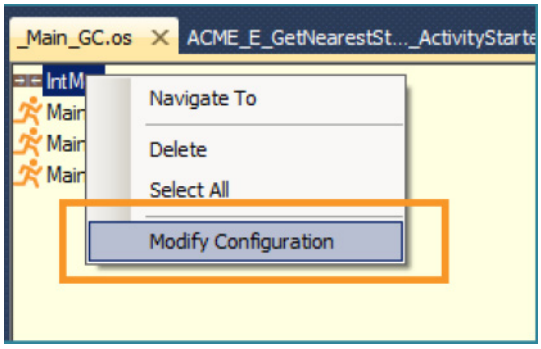
Adding Context Values to the 360 App Bar

Follow these steps to add context values to the 360 View primary view so it will display additional customer information. To do so, you will make modification to the Interaction.xml | Primary section.

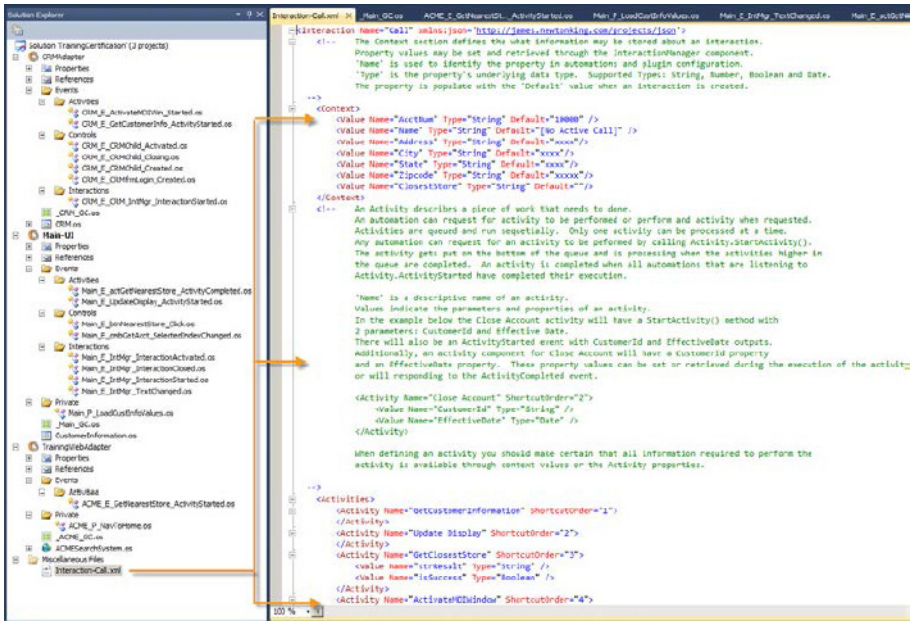
Note Make a back-up copy of the Interaction.xml file before you modify the .xml file. You can modify the .xml file via OpenSpan Studio.

To make changes to the interaction.xml file in Studio, follow these steps.

1. Open your solution created. Double-click the Main Global Configuration item in Solution Explorer.
2. Right-click the applicable item and click Modify Configuration. You can also double-click the Interaction.xml item in Solution Explorer.



The Interaction.xml file opens in the Designer window.



3. Add the additional context values to the 360 App bar plugin section.
4. Return to the Interaction.xml file in Solution Explorer.

5. In the Plugin name | Primary section, add these context value lines after the <Label>Account Number</Label> line. Here is an example:

```
</Item>
<Item name="State">
    <Context Name="State" />
    <Label>State</Label>
</Item>
<Item name="Zipcode">
    <Context Name="Zipcode" />
    <Label>Zipcode</Label>
```

6. Select File | Save All.

Note Before testing the modification to the Interaction.xml file, you must choose the Build | Clean Solution option every time a change is made to the Interaction.xml section and also before you debug the solution.

Adding Context Values to the 360 App Bar Extended View

Currently, the Agile Desktop 360 View Extended section displays the Address, City, State, and ZIP Code. To add an additional contexts to the 360 View extended view, you modify the Interaction.xml | Extended section.

Follow these steps:

1. In the Plugin name | Extended section, add the additional context value lines after the <Label>Zipcode</Label> line. Here is an example:

```
<Item name="ClosestStore">
    <Context Name="ClosestStore" />
    <Label>ACME Nearest Store</Label>
```

2. Select File | Save All.

Styling Items in the 360 View

You can style the items on the 360 View to draw attention to alerts, add emphasis, and assign a rank to the items that appear. You can list multiple formatting options for each item that appears via the plug-in. The system uses the first formatting option that matches.

The Formatting element has these attributes:

- `<Formatting Operator="op" Value="value" Use="display style" />`

  The item's value (context or global) is compared to the value specified in the Formatting element. For instance, if an item uses a global value, the global value is used. The values are also evaluated based on the operator. You can include any of the True/False expression comparison operators. For more information, see "Operators" on page 24.

- `<Formatting Criteria="True/False expression" Use="display style" />`

  The system evaluates the True/False expression you specify via the Criteria attribute. The expression can reference multiple context values as well as global values.

- `<Formatting Use="display style" />`

  A Formatting element without Operator or Criteria attributes will always match. You can use this, for instance, to tell the system that if none of the Formatting attributes match, use a specific display style. Keep in mind the system ignores the Formatting elements for an item placed *after* this type of element.

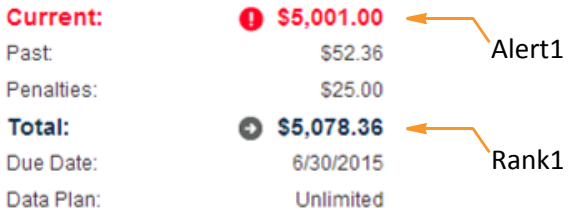
Here is an example:

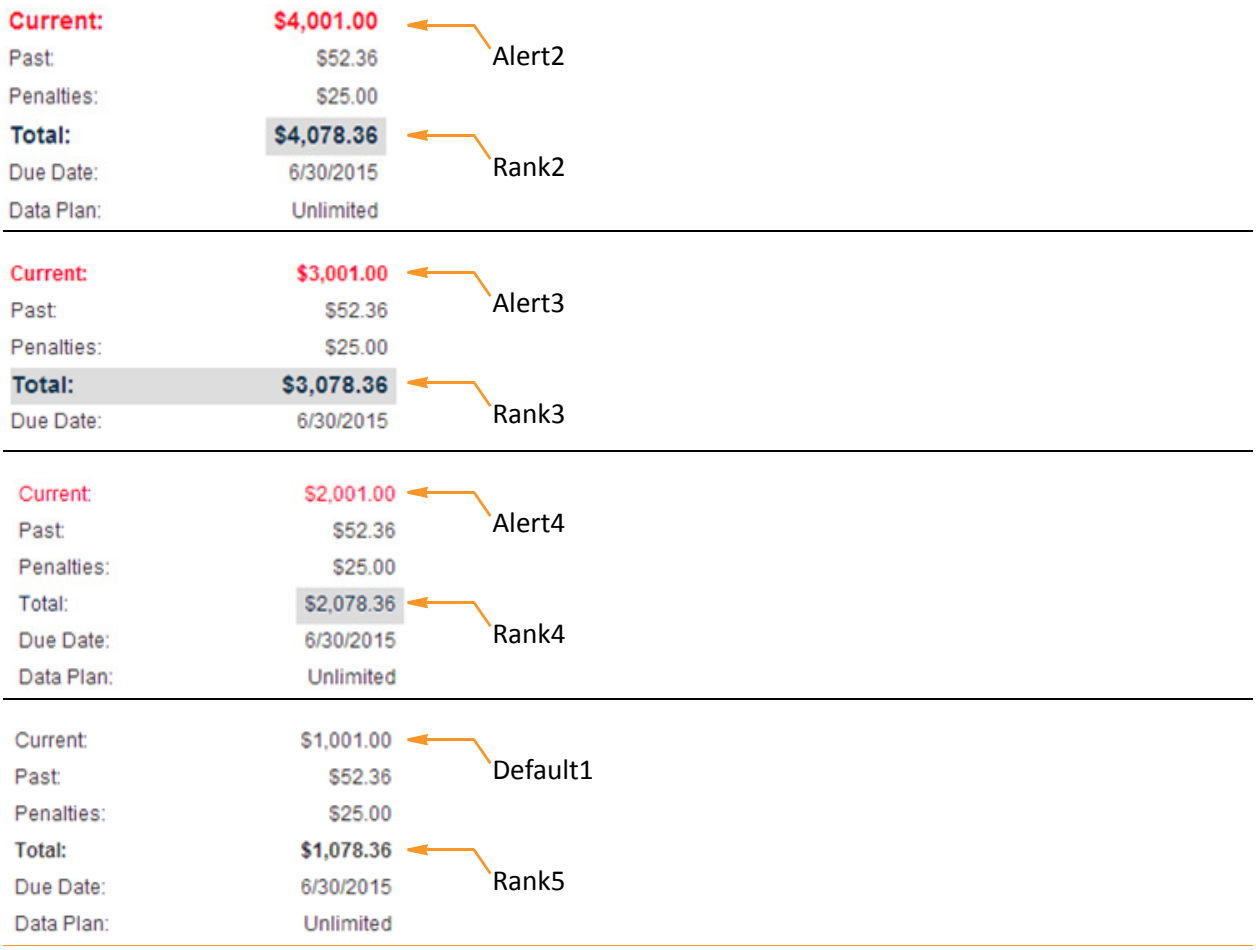
```
<Item name="Balance">
   <Context Name="Balance" />
   <Label>Balance</Label>
   <Formatting Operator="gt" Value="1000000" Use="Alert1" />
   <Formatting Criteria="Balance lt 100 And AccType eq 'Checking'" Use="Alert2" />
   <Formatting Use="Rank1" />
</Item>
```

This example tells the system to use the Alert 1 style for the Balance amount if the amount is greater than $1,000,000 and use the Alert 2 style if the Balance is less than $100 and the account type is checking.

Display Styles

Here are examples of the display styles:





Syntax

Here is the syntax of a True/False expression:

```
operand [operator operand]
```

Operands

An operand can be one of the following:

- A string literal. Delimited by double (“) or single (‘) quotation marks.
  - “Double quoted string literal”
  - ‘Single quoted string literal’
- Numeric literal. Digits with, optionally, a single decimal point.
  - 1000
  - 999.99
- Boolean literal. One of these keywords:

  - True
  - False
- Variable. Corresponds to the name of a Context or Global value. Use these prefixes to specify which value to use:
  - Interaction.
  - Globals.

  If you omit the prefix, the system assumes it is a Context value.
- A True/False expression in parenthesis.

Operators

The operator can be one of the following:

- Comparison operators:
  - Eq — Equals
  - Lt — Less than
  - Gt — Greater than
  - Le — Less than or equal to
  - Ge — Greater than or equal to
  - Ne — Not equal to
  - StartsWith — The string on the left starts with the string on the right.
  - Contains — The string on the right is contained within the string on the left.
- Relational operators:
  - And
  - Or

Examples

Here are some examples:

These examples show how the use of parentheses can change the meaning:

```
Criteria="AccountType eq 'Checking' and (Balance gt 1000 or Balance lt 50)"
Criteria="(AccountType eq 'Checking' and Balance gt 1000) or Balance lt 50"
```

Assuming that you defined HasDataPlan as a Boolean, these examples return the same result:

```
Criteria="HasDataPlan"
Criteria="HasDataPlan eq true"
Criteria="HasDataPlan ne false"
```

Comparing a date value with a literal:

```
Criteria="TransactionDate lt '12/25/2015'"
```

This example compares two values and uses the value location prefixes:

```
Criteria="Interaction.TransactionDate gt Globals.CutoffDate"
```

Making 360 View Items Clickable

You can set up any item on the 360 View to be a clickable shortcut, if you are using OpenSpan version 7.1.63 or higher. Use the following Interaction.xml file Plugin/Section/Item: options to make an item clickable:

Option	Description
ShortcutType	Use this option to specify the type of shortcut. Choose from these types of shortcuts: • Activity - Starts an activity. • Url - Goes to a web page. • External - Starts an executable. • Automation - Starts an OpenSpan automation.
ActivityName	If this is an Activity shortcut, enter the name of the activity that you want to start when the user clicks this item.
Target	Enter the destination for the shortcut. For example, if the shortcut is a web site, you enter the URL. If the shortcut launches an executable file, enter the name of that file.
Project	If this is an automation shortcut, enter the name of the project that contains the automation. If the project name contains a space, replace the space with an underscore. For instance, *My Project* should become *My_Project*.
AutomationName	For automation shortcuts, enter the name of the automation that you want to start when a user clicks the item. If the automation name contains a space replace, it here with an underscore. For instance, *My Automation* should become *My_Automation*. **Note**: The automation should have an entry point and you can pass parameters into the entry point if needed.
Params	Use this option to pass parameters into automation. Only use this option if you chose Automation as the Shortcut type. The parameters should be a comma-delimited list that is represented as a single string. Keep in mind that the parameters are passed into the automation in the order as they appear in the list. You can use these keywords inside the params string: • Context. – Prefix with Context. to use values defined in the Context section of this document. The item passed will be that of the active interaction. • Globals. – Prefix with Globals. to use values defined in the Globals section of this document. • Interaction.ActiveKey – Use this to pass in the current active interaction key. The parameters will be listed as a string and can be enclosed in either quotation marks (“ ”) or apostrophes (‘ ’). If you are passing a string literal as a parameter to an automation, then it should be enclosed in apostrophes and the parameter list would then be enclosed in quotation marks, or vice versa.

Here are some examples of correct params usage:

Params string in Interaction.XML	Automation entry point 1	Automation entry point 2
“Context.FullName, ‘John’”	The FullName context value of the active Interaction as a string	The string: *John*
‘Context.FullName, “John”’	The FullName context value of the active Interaction as a string	The string: *John*
“’15’, ’John#apos;s’”	The number: *15*	The string: *John’s*

Params string in Interaction.XML	Automation entry point 1	Automation entry point 2
“Globals.AgentName, ’#quot;John#quot;’”	The AgentName value as declared in the Globals section as a string	The string: *“John”*

Note Use regular HTML entities for other special characters if there is a XML syntax error. Also note that if there is no active interaction, *Context*. and *Interaction.ActiveKey* are passed as null values.

Adding Items to the Shortcut View

These steps show you how to add functionality in the Shortcut section.

The advantage of adding this functionality is that the user no longer needs to have the Customer Information window in focus to retrieve the values from the automation. The Shortcut button on Agile Desktop can instead perform the action.

In this topic, you will see two ways to change the configuration that will simulate the same action. The main difference between these configurations is:

- **Method 1**: Adds a context value that queues the activity first before it runs the automation.
- **Method 2**: Adds a context value that starts the automation via an Execute method (Entry point) and starts the interaction without the activity being queued beforehand.

In both cases, you will make modification to the Interaction.xml | Shortcut section.

Method 1: Adding the configuration change to queue the activity, then running the automation

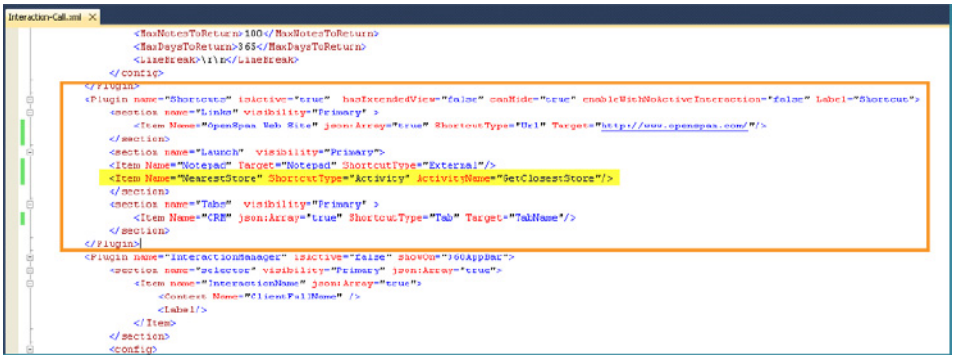
Follow these steps:

1. Open the Interaction.xml file.
2. In the Plugin name | Shortcuts section, add the shortcut. For instance, to add an activity named GetClosestStore you would:

   <Item Name=”NearestStore“ ShortcutType=”Activity” ActivityName=”GetClosestStore”/>

   after the <Item Name=”Notepad” Target=”Notepad” ShortcutType=”External”/> line.

   The Interaction.xml |Shortcuts section would look as follows:



3. Select File | Save All.

Method 2: Adding the configuration change to the Shortcut section to launch the automation without first queueing the activity

Follow these steps:

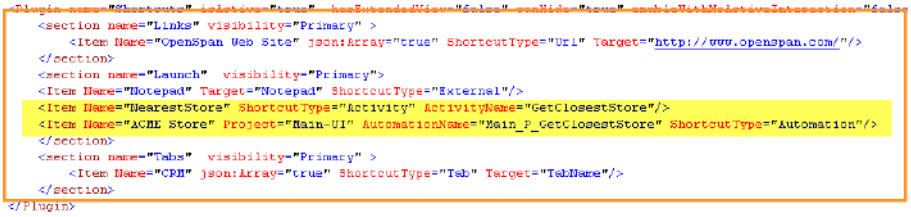
1. Open the Interaction.xml file. In the Plugin name | Shortcuts section, after the following section:

   <Item Name=”NearestStore” ShortcutType=”Activity” ActivityName=”GetClosestStore”/>

   add the following:

   <Item Name=”ACME Store” Project=”Main-UI” AutomationName=”Main_P_GetClosestStore” ShortcutType=”Automation”/>

   The Interaction.xml | Shortcuts section should look as follows:



2. In Solution Explorer, right-click the Private folder in the Main-UI project and choose Add | New Automation.
3. Assign a name the automation and set the ShowDesignCompNames attribute to True.
4. Right-click in the white space of the automation and select Add Entry Point.
5. Add the necessary design blocks.
6. Select File | Save All.

Enabling and Disabling Shortcuts

If you are using OpenSpan version 7.1.63 or higher, you can use the Enable element as a sub element of both the Item and MenuItem elements in the Interaction.xml file to add a criteria expression that enables or disables a shortcut. Here is an example:

```
<Item Name="Validate Caller" ActivityName="ValidateCaller" ShortcutType="Activity">
   <Enable Criteria="Globals.MaxCharges gt Interaction.CurrentCharges" />
</Item>
```

If the criteria expression evaluates to True, the button is enabled, otherwise the button is disabled. If you omit the Enable element, the button is enabled.

Note For more information on creating the criteria expression, see “Styling Items in the 360 View” on page 22.

Keep in mind...

- You can base the criteria expression on Interaction Context or Global Dictionary items.
- Disabled buttons are greyed out.
- If a button is disabled, click events will not fire for the button.

Determining Which Shortcuts Appear

If you are using OpenSpan version 7.1.63 or higher, you can use the Visibility element to specify if a shortcut button will be visible based on the results of a logical expression. You can use context values and global dictionary items. For example, for showing and hiding buttons which lead to updating customer records, you could include a Boolean interaction context value called *ValidatedCaller* and use one of the following:

- `<Visible:Criteria="ValidatedCaller"/>`
- `<Visible:Criteria="ValidatedCaller Eq 'True'"/>`

The default Criteria will evaluate to True and will be visible.

Here is an example:

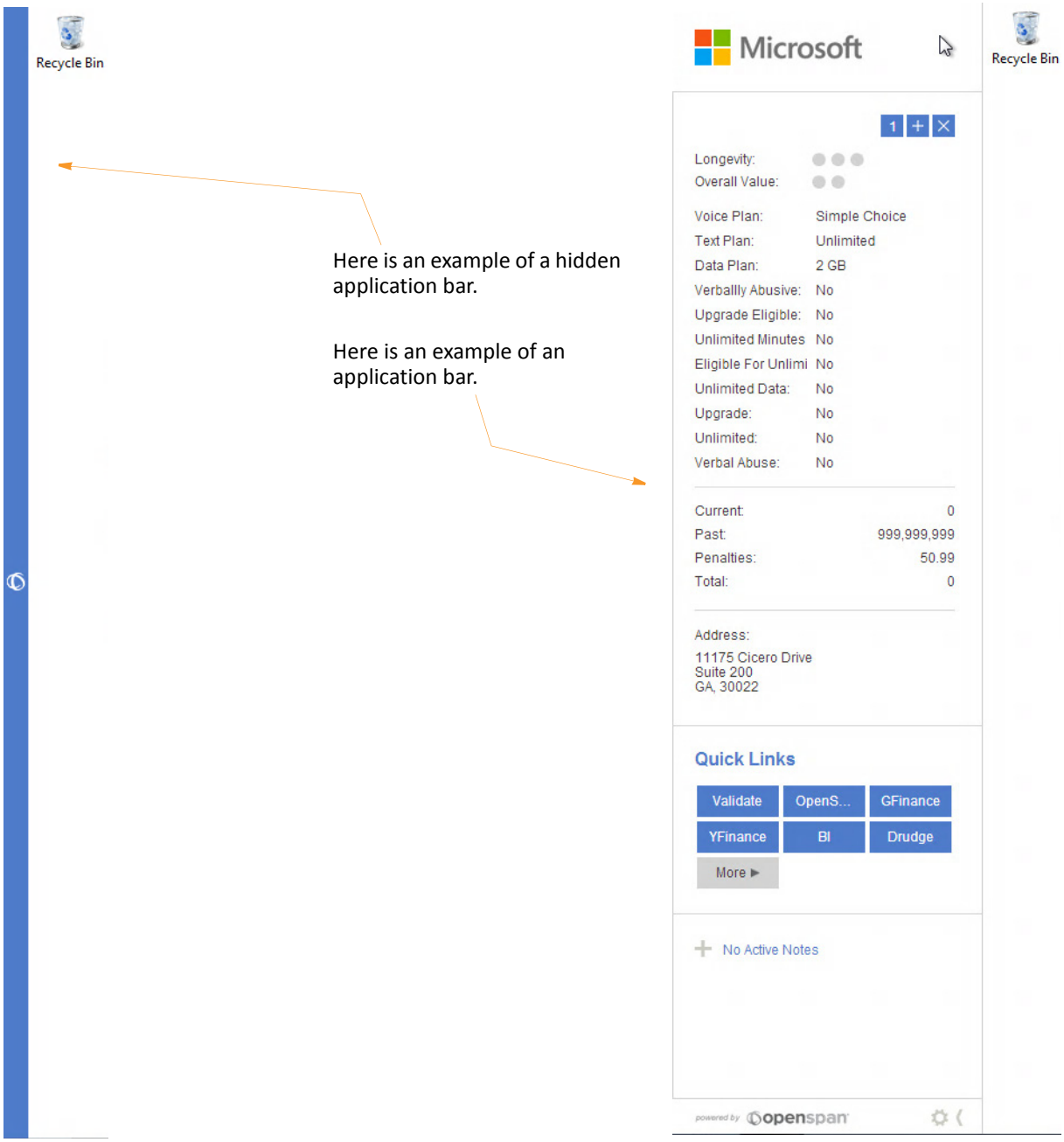
Note Be careful when using this option. Buttons that appear and disappear can confuse users.

Note too the difference between enabling and disabling shortcut buttons and hiding or showing them. Disabled buttons are simply grayed out but retain their position in the list of shortcuts. Hidden buttons do not appear and do cause the list of shortcuts to reflow.

```
<Section Name="Links" Visibility="Primary">
   <Item Name="Validate Caller" ActivityName="ValidateCaller" ShortcutType="Activity"/>
   <Item RowOneLabel="OpenSpan" RowTwoLabel="Web Site" Target="http://www.openspan.com/"
   ShortcutType="Url">
      <Visible Criteria="SocialInfluence eq 0" />
   </Item>
```

Defining the Application Bar

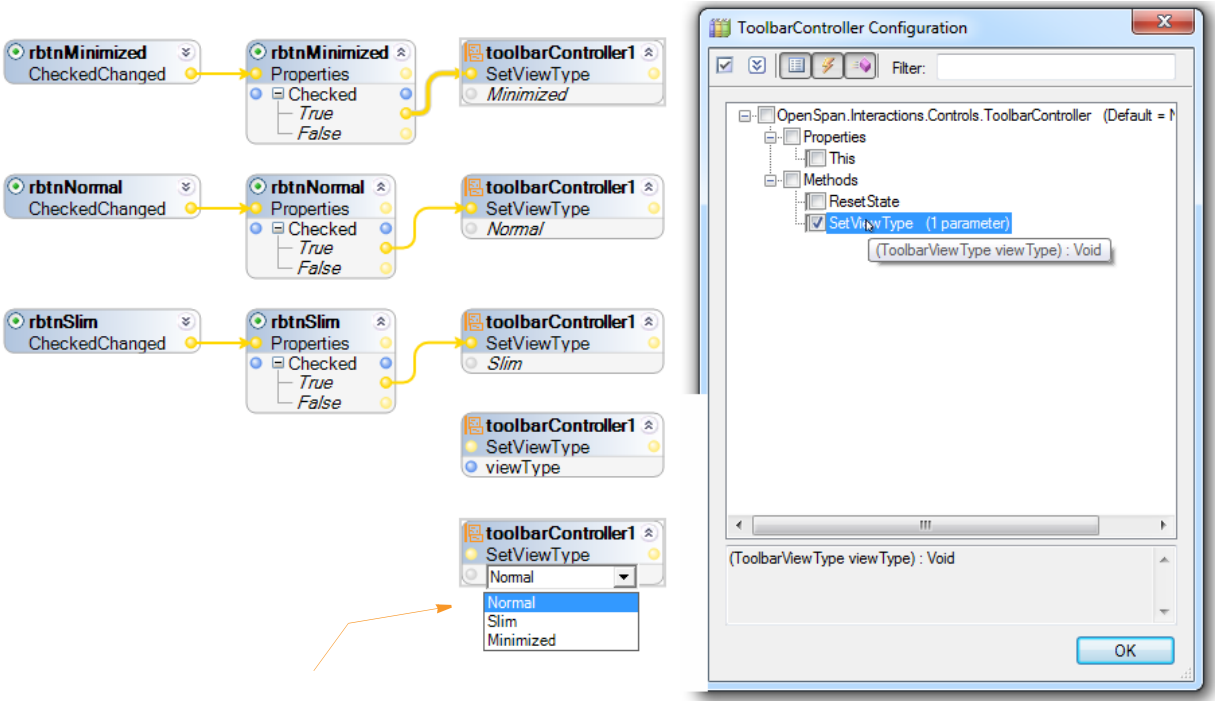
The main component of the user interface is the Agile Desktop sidebar, a thin, 275-pixel, bar positioned on the left edge of the primary monitor. The sidebar lets end users hide or expand its contents. Here is an example:



Determining the View

Build 7.1.34.0 adds the toolbarController component, which you can use to control the display of the AgileDesktop toolbar. This component lets you create solutions which can switch between the normal, slim, and minimized views.

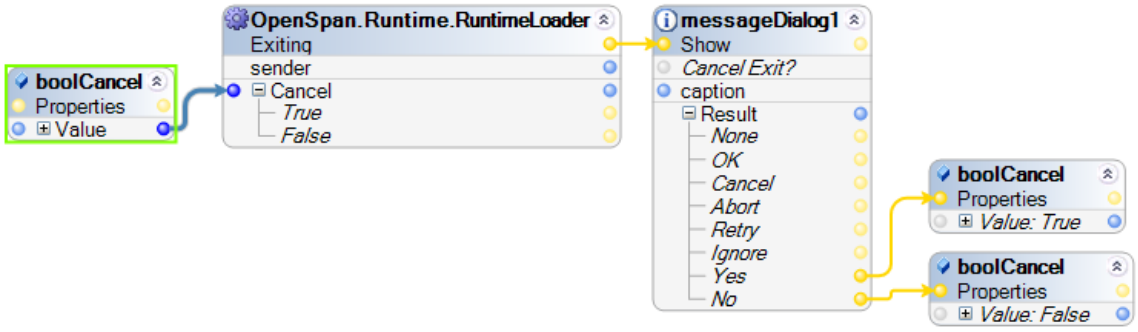
Use the SetViewType method to specify the view you want. Here is an example:



Controlling Shutdown

Build 7.1.24.0 of Agile Desktop added a static Exiting cancellable event to the RuntimeLoader. You can use this event in an automation to present a confirmation dialog and, optionally, prevent Agile Desktop or Runtime from shutting down.

Here is an example of how it works:



Adding Plug-ins

Agile Desktop includes these plug-ins:

Using the 360 View Plug-in

Once an Agile Desktop user is logged into all of the required applications so he or she can begin taking calls, navigating those systems to get the information the customer is requesting can be difficult and time-consuming. For instance, a customer can have multiple accounts, such as accounts for banking, checking, credit cards, and mortgage. The data for each of these may reside on different systems.

A 360 view gathers all pertinent information in a single pane, so the Agile Desktop user can quickly provide the information the customer is requesting. Having this consolidated view of customer data increases the productivity of the end user and increases customer satisfaction by reducing wait time.

By eliminating the need to toggle between systems and navigate layers of screens, you give the Agile Desktop user more time to focus on higher-value activities with the customer, such as offering new promotions or programs which can result in additional revenue opportunities.

Agile Desktop, provides a very basic, non-intrusive way to aggregate key customer data quickly and flexibly. Using Agile Desktop, you can pilot additions to the 360 view with smaller, more controlled user groups, and then roll the additions to the broader user population once the additions are finalized.

The 360 View plug-in has these views:

- Primary view. This is always visible in Agile Desktop.
- Extended view. This is displayed by clicking on the on-hover arrow in the primary view. The information that displays is related to the plug-in from where the extended view is opened.

The initially defined sections are as follows:

Section	Description
Identity	Displays, for example, the caller's name.
Ratings	Each value in this section is represented as a bar of dots from 1 to 5 dots.
Products	Displays the products that pertain to the active caller.
Balances	Displays monetary values like current balances.
Eligibility	A text value.
Information	Displays contact information such as address, telephone, and email.

Here is an excerpt from the default definition. Note that the name of the plug-in is *360AppBar*.

```
<Plugin name="360AppBar" isActive="true" hasExtendedView="true" canHide="false" enableWithNoActiveInteraction="false">
  <section name="Identity" visibility="Primary">
    <Item name="FullName" json:Array='true'>
      <Context Name="ClientFullName" />
      <Label></Label>
    </Item>
  </section>
  <section name="Ratings" visibility="Primary">
    <Item name="SocialInfluence">
      <Context Name="SocialInfluence" />
      <Label>Social Influence</Label>
    </Item>
    <Item name="LifetimeValue">
      <Context Name="LifetimeValue" />
      <Label>Lifetime Value</Label>
    </Item>
  </section>
  <section name="Products" visibility="Primary">
    <Item name="VoicePlan">
      <Context Name="VoicePlan" />
      <Label>Voice Plan</Label>
    </Item>
    <Item name="TextPlan">
      <Context Name="TextPlan" />
      <Label>Text Plan</Label>
    </Item>
    <Item name="DataPlan">
      <Context Name="DataPlan" />
      <Label>Data Plan</Label>
    </Item>
  </section>
  <section name="Balances" visibility="Extended">
    <Item name="Current">
      <Context Name="CurrentCharges" />
      <Label>Current</Label>
    </Item>
    <Item name="Past">
      <Context Name="PastDue" />
      <Label>Past</Label>
    </Item>
    <Item name="Penalties">
      <Context Name="Penalties" />
      <Label>Penalties</Label>
    </Item>
    <Item name="Total">
      <Context Name="TotalDue" />
      <Label>Total</Label>
    </Item>
  </section>
  <section name="Eligibility" visibility="Extended">
    <Item name="Upgrade">
      <Context Name="EligibleForUpgrade" />
      <Label>Upgrade</Label>
    </Item>
    <Item name="Unlimited">
      <Context Name="EligibleForUnlimited" />
      <Label>Unlimited</Label>
```

Adding Whitespace Around Images

By default, the system adds padding around any images you use in the 360 View plug-in. Beginning with version 7.1.63, however, you can customize the amount of padding using this Interaction/xml file option:

`Plugin/Section:ImagePadding`

Enter a number which specifies, in pixels, the amount of padding to add. This option only affects the Images section. If you leave this option blank, the default padding of 24 pixels is used.

Using the Notes Plug-in

Call wrap-up can be the most time-consuming part of the end user/customer interaction. During customer interactions, OpenSpan can monitor agent activity and automatically post events into notes. For instance, if the Agile Desktop user offers a special promotion, OpenSpan can capture the offer type and document the customer's acceptance or refusal.

During a customer interaction, Agile Desktop users often have to track a great deal of information. Sometimes they resort to writing account numbers on note pads, jotting deposit amounts on sticky notes, or repeating transaction codes over and over while they navigate through numerous screens to reach the field where the data must be entered. These practices are inefficient and error-prone and may even breach compliance and data security guidelines.

By automating call wrap-up, you can capture all critical data points and notate the account without the Agile Desktop user having to manually document each step of the interaction. This saves time, improves accuracy, and increases productivity.

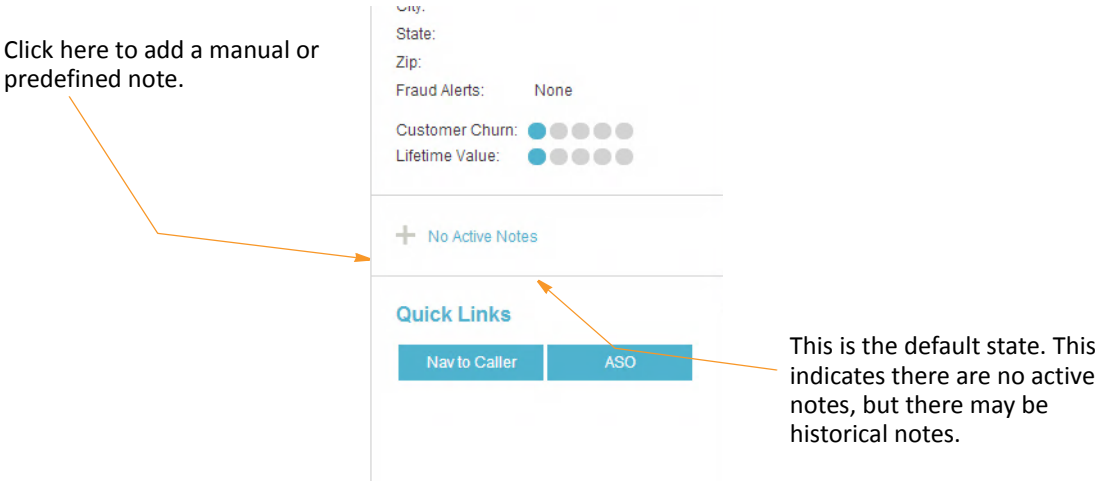
With this in mind, the Notes plug-in makes it easy to enter these types of notes:

- Manual — Notes are simply notes typed in by the Agile Desktop user.
- Predefined — The text for predefined notes is defined once and inserted as often as is necessary. The Agile Desktop user can edit this text after it is inserted. Typically, predefined notes are standard sentences or phrases used over and over. Agile Desktop users simply select the text from a list, instead of typing it into the text area.

The Notes plug-in has these views:

View	Description
Primary	Always visible in Agile Desktop. Shows a list of active notes, with one line per note. Each note is hyperlink that when clicked open the note in edit mode. Only shows the accumulation of recent notes, not historic notes.
Extended	Visible when you click the arrow in the primary view. Shows notes in descending chronological order. An automation retrieves the notes from the system of record. Also shows the date and time the note was created as well as three lines of every note. You can expand and scroll each note to see the entire note. You can also scroll through the list of notes.

Here is an example:



This is the primary view when there are no active notes. There are no active notes when you start an interaction. Here are the characteristics of the Notes plug-in:

General

- When there are no active notes, the Notes plug-in area of the primary view displays the static text *No Active Notes* and a Plus (+) icon or button. You click this icon to add a manual or predefined note.
- If you delete all active notes during an interaction, the *No Active Notes* text and Plus (+) icon display again.
- Notes added manually or by selecting predefined text replace the *No Active Notes* text and the Plus (+) icon with new note controls.
- Historical notes, which display only in the extended view, do not affect what appears in the primary view Notes plug-in area.

The Predefined Notes list

When you select a value from the Predefined Notes list, that text is added to the text area. You can add as many predefined notes as you need.

Specifying the Predefined Note Text

Edit the NoteTemplates section of the Interaction.xml file to add or edit predefined text. Here is an example:

```
<Plugin name="Notes" isActive="true"  hasExtendedView="true"  canHide="true" enableWithNoActiveInteraction="false">
  <section name="NoteTemplates" visibility="None" json:Array='true'>
    <Item name="CallStarted" >
      <Content>Call with {{ClientFullName}} started</Content>
    </Item>
    <Item name="InformDueAmount">
      <Content>Informed of {{TotalDue}} due amount {{#EligibleForUpgrade}} and upgrade eligibility{{/EligibleForUpgrade}}</Content>
    </Item>
  </section>
```

Option	Description
Item name	This is a name assigned to a predefined note.
Content	Enter the text that comprises this predefined note. You can insert variables, such as TotalDue and ClientFullName. For more information, see "Using Variables in Predefined Notes" on page 37.

Using Variables in Predefined Notes

You can use variables when defining the content of a predefined note. For instance. you could insert the variable ClientFullName as shown here:

```
<Content>Call with {{ClientFullName}} started</Content>
```

If the client's name was Ed Ricketts and that value was stored in the Contexts section in ClientFullName, you would get this predefined text:

Call with Ed Ricketts started

These variable values are temporarily stored by the Interaction Host and must be specified in the Context section of the Interaction.xml file. The Interaction Host gets the values via an automation which in turn may get the values from an external program. Be sure to enclose variables inside double braces, as shown here:

```
{{variable}}
```

Some of the variables defined in the Contexts section of the Interaction.xml file store Boolean values, such as EligibleForUpgrade. You can use these Boolean variables to set up conditional phrases, such as

```
<Content>Informed of {{TotalDue}} due amount {{#EligibleForUpgrade}} and
upgrade eligibility{{/EligibleForUpgrade}}</Content>
```

The text between the hash tag (#) and the slash (/) is conditional, based on the value of EligibleForUpgrade. In this example, if TotalDue amount is $100 and EligibleForUpgrade is True, the predefined note would read...

Informed of $100 due amount and upgrade eligibility

If TotalDue amount is $100 and EligibleForUpgrade is False, the predefined note would read...

Informed of $100 due amount

Here is the complete example, showing all of the options from the Interaction.xml file:

```
 <Item name="CallStarted" >
   <Editable>False</Editable>
   <Content>Call with {{ClientFullName}} started</Content>
  </Item>
  <Item name="InformDueAmount">
    <Editable>False</Editable>
    <Content>Informed of ${{TotalDue}} due amount {{#EligibleForUpgrade}} and
upgrade eligibility{{/EligibleForUpgrade}}</Content>
  </Item>
```

In this example, the text *Call with Ed Ricketts started* would appear in the drop down for predefined notes. If you were on a call with a user named Ed Ricketts and you chose CallStarted, Agile Desktop would insert this note:

Call with Ed Ricketts started

You would not be able to edit this predefined note, based on the value for the Editable option.

If you then chose to insert the InformDueAmount predefined note and Ed Ricketts owed $100, Agile Desktop would insert this note:

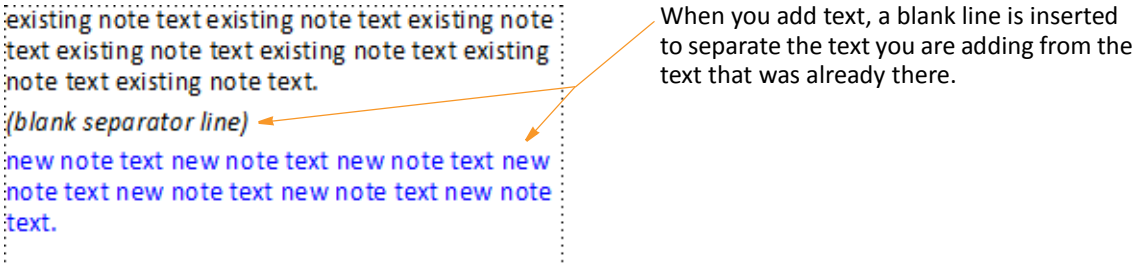
Informed of $100 due amount and upgrade eligibility

Note For more information on defining the contexts in the Interaction.xml file, see this topic in OpenSpan Help:

Using the Interaction.xml File

Keep in mind...

- Users do not have to select notes from this list.
- Selecting a value from the list populates the text area with the text associated with that value and not the list value itself, although they can be the same.
- If you type text in the text area and then select an item from the list, the item text from the list is added below the text you typed.
- Each predefined note begins on a new line, which follows a blank line that is added below a manual note.
- If you select more than one item from the list, each item is added to the text area as a new line, separated by a blank line. Here is an example:



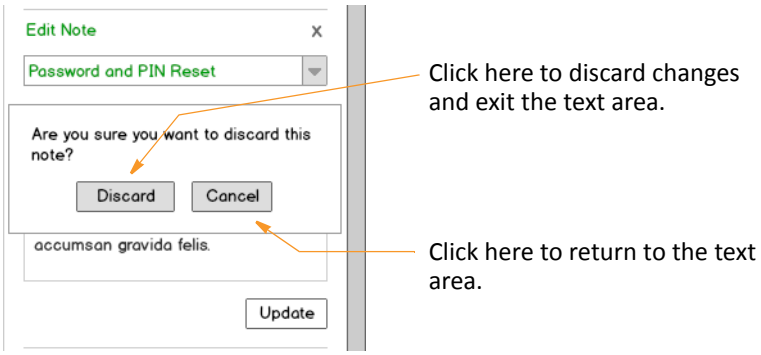
Text area

The text area is where you enter text or where the predefined text you select appears. It appears after you click the Plus (+) icon. If you select predefined text, once that text is in the text area, you can edit it. Changes do not affect the predefined text definition, only the content of that particular note.

- An entry is required in the text area. The Add or Update button is disabled until you make an entry into the text area either by typing text or by selecting predefined text.
- There is no limit to the number of characters you can enter.

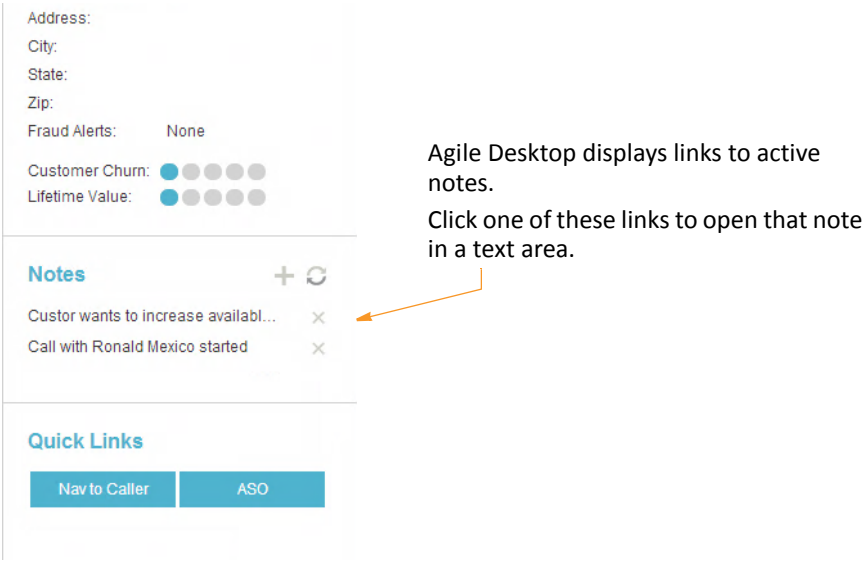
Note Clicking the Add button stores the note *locally* — it does not save it in the system of record — and places the new note at the top of the Active Notes list.

Here is an example:



Active Notes list

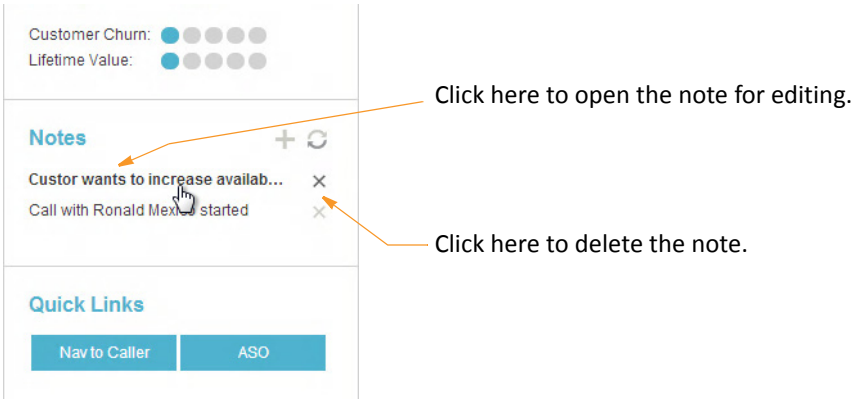
Links to active notes appear in a list. Click a link to see and edit the actual note text. For example, if you want to edit a note, you click on its link and the note opens in the text area. Here is an example of the default view when there are active notes:



- The plug-in displays the first 35 characters each note, including spaces, followed by an ellipsis (...).
- The Sync button...
  - Saves all active notes to the system of record.
  - Changes active notes into historic notes and displays them on the extended view.

Editing a note

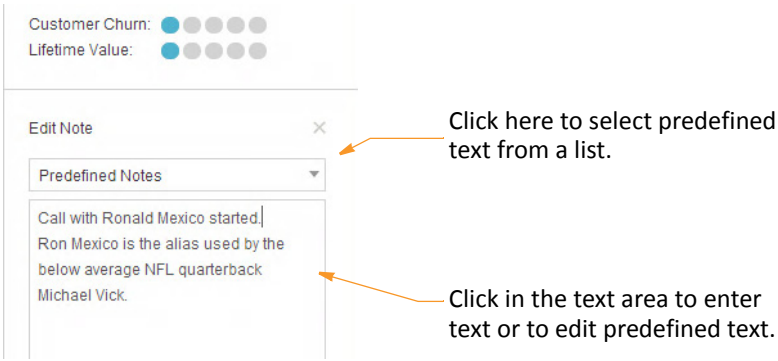
Every active note is a hyperlink. If you hover the mouse pointer over an active note and highlight the note, the mouse pointer changes to a hand, and an X (Delete) icon displays at the end of the row. Here is an example:



Clicking an active note link replaces the active note list with the Edit Note controls.

Editing Note controls

Here is an example:

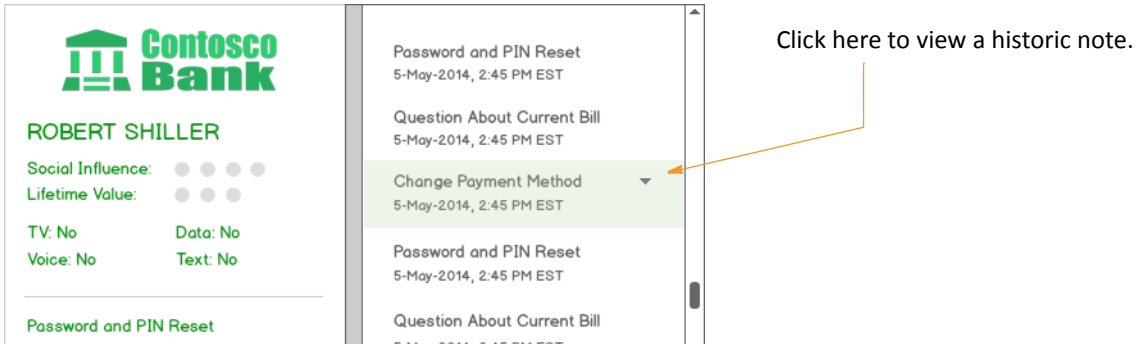


Keep in mind...

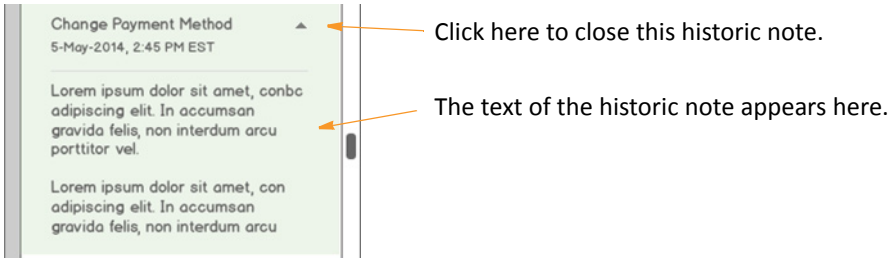
- All historic notes display on the extended view. Use the scroll bar to scroll through the historic notes.
- Historic notes are read-only.
- Historic notes show the time and date the note was last synced.
- Historic notes appear in chronological order, with the newest notes first.
- The name of the user who created the note does not display.

Viewing historic notes

To view a historic note, highlight the note and click the down chevron, as shown here:



After you click the down chevron to view a historic note, the note opens and the chevron changes to an up chevron, as shown here:



Synchronizing Notes

All notes are active until you sync (save) them. This activity, which by default is omitted from the Notes plug-in, saves the note to the system of record. Use an automation to drive this activity, meaning you can use the Agile Desktop UI to manually add notes or you can have an automation add the note. Either way, the notes are active until they are synced.

The automation should be triggered when the user clicks Sync saves them to the system of record. When the automation then retrieves the saved notes to refresh the Agile Desktop UI, those notes — that were active a minute ago — are now in the historical notes.

Use the OpenSpan Notes Controller component in an automation to access the Agile Desktop Notes plug-in. There are several methods and events which are specifically applicable to this component. For more information, see this help topic: Using the OpenSpan Notes Controller.

Note Think of the Agile Desktop UI as a vehicle for adding notes and displaying historical notes.

Notes Activities

The Notes plug-in expects and uses these activities:

```
<Activity Name="LoadHistoricNote">
   <Value Name="NoteText" Type="String" />
   <Value Name="NoteDate" Type="Date" />
</Activity>
```

The plug-in listens for this activity and every time it is started, places the note into the historic list.

```
<Activity Name="AddNote">
    <Value Name="NoteText" Type="String" />
    <Value Name="Predefined" Type="Boolean" />
</Activity>
```

The plug-in listens for this activity. Every time it is started, the plug-in places the note in the active notes list. The new note appears in both the Standard and Extended view. If the Predefined parameter is True, the NoteText is treated as the name for a note template, otherwise the NoteText is the note itself. Either way, the note is processed for replacement tokens.

```
<Activity Name="SaveNotes">
    <Value Name="Notes" Type="String" />
</Activity>
```

The plug-in starts this activity when the Save button is clicked, passing all the active notes. The Notes parameter is a JSON string containing an array of notes, as shown here:

```
{
  "Notes":
  {
    "Note":
    [
      {
        "NoteDate": "2013/12/03 13:46:00",
        "NoteText": "Call with Ed Ricketts started"
      },
      {
        "NoteDate": "2013/12/03 13:48:00",
        "NoteText": "Informed of $100.00 due amount"
      }
    ]
  }
}
```

Predefined Notes Definitions

The predefined note text is defined in the Notes plug-in section of the Interaction.xml file. Here is an example:

```
<NoteTemplates>
    <NoteTemplate Name="CallStarted" Editable="False">
        Call with {{ClientFullName}} started
    </NoteTemplate>
    <NoteTemplate Name="InformDueAmount" Editable="False">
        Informed of {{TotalDue}} due amount
   {{#EligibleForUpgrade}}
   and informed of upgrade eligibility
   {{/EligibleForUpgrade}}
    </NoteTemplate>
    <NoteTemplate Name="Empty" Editable="True"></NoteTemplate>
</NoteTemplates>
```

Replacement Tokens

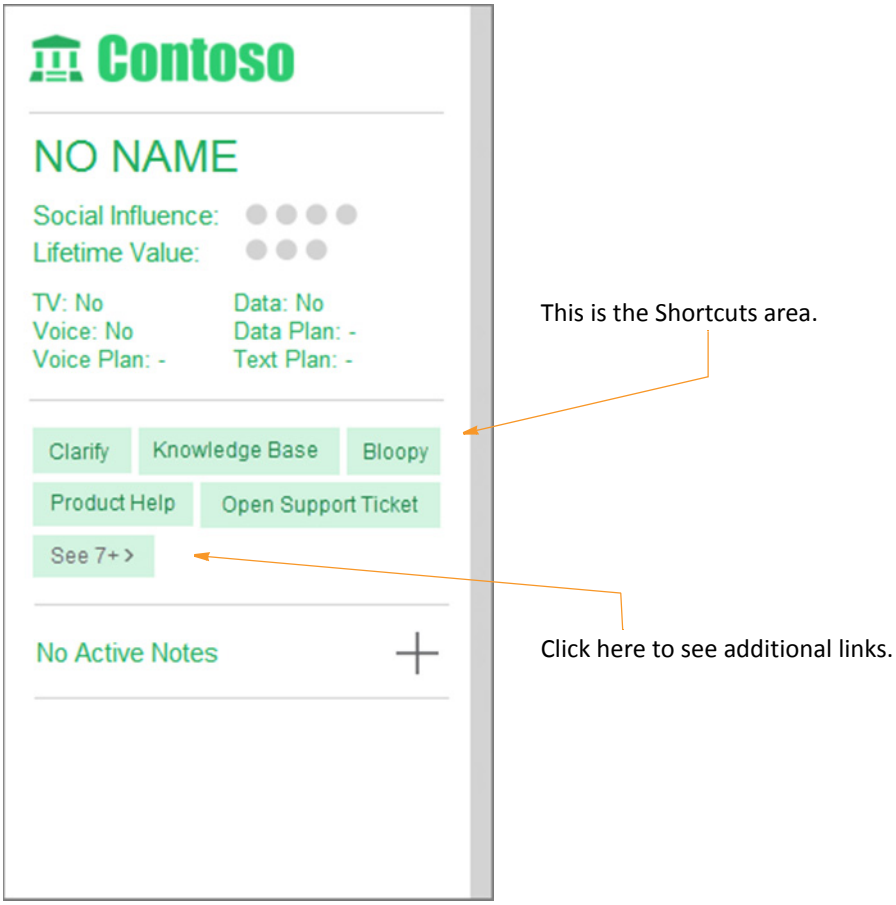
When adding a note comprised of either predefined or free form text, the text is processed for token replacement. The token replacement is based on Context values and follows a Mustache style. These forms are supported:

Form	Description
Context value replacement	If the text contains the name of a context value enclosed in double curly braces, this token is replaced with the current value of the context value. Here is an example. If the Note text is: `Call with {{ClientFullName}} started` And the current value of ClientFullName is *Ed Ricketts,* the result is: *Call with Ed Ricketts started*

Conditional inclusion	If the text contains a start and end section marker, that text is conditionally included. A section marker has this format:
	`{{#context-name}}text{{/ context-name}}`
	Here is an example. If the Note text is:
	`Informed of ${{TotalDue}} due amount`
	`{{#EligibleForUpgrade}}`
	`and upgrade eligibility`
	`{{/EligibleForUpgrade}}`
	And the current value of
	• TotalDue: $100.00
	• EligibleForUpgrade: True
	Then the result is:
	Informed of $100.00 due amount and upgrade eligibility

Using the Shortcuts Plug-in

The Shortcuts plug-in lets you add a series of shortcut buttons to Agile Desktop to launch applications or go to web sites. These buttons appear under the Quick Links heading in the Agile Desktop user interface. Here is an example:



Types of Shortcuts

Shortcuts are categorized into these types:

Type of shortcut	Description
Activity	Starts an activity, based on the activity name you specify.
URL	Launches a browser window with the URL you specify.
OpenSpan	Starts an OpenSpan automation package.
External	Launches an external program not controlled by an OpenSpan automation.
Tab	Positions the given tab in the reparenting area.

Note You cannot use a shortcut button to start a task. You can, however, use a shortcut button to start an automation which starts a task.

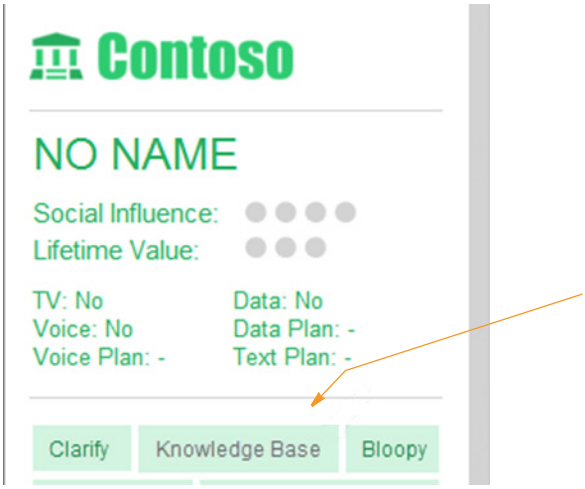
If the shortcut causes a program to appear in the reparenting area, it is outside the shortcut's control. For example, an Activity shortcut can cause automation packages to launch and, if they are configured for reparenting, those packages appear in the reparenting area. The shortcut itself, however, simply starts the activity.

Shortcut Characteristics

Shortcuts have these characteristics:

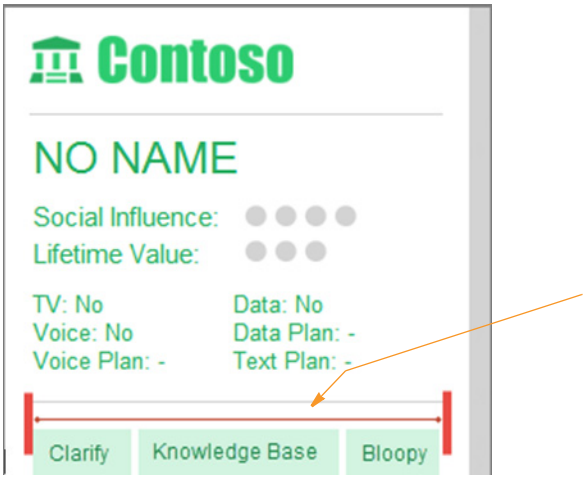
- Shortcuts appear as buttons.
- Buttons are always enabled.
- Buttons are sized automatically, based on the length of the button label text.
- Margin/padding, right, left, top, and bottom, is the same for all buttons.
- The order in which buttons are listed in the configuration file defines the order in which the buttons appear in Agile Desktop.
- For the background color, the normal state button background is a 25% saturation of the accent color you defined. You can use this web site to get color hex values:

  http://colorschemedesigner.com/csd-3.5/
- For text, the normal state button label font color is the accent color — except for the More/Less button.
- The appearance change during a hover (rollover) and active (during clicking action) state is limited to the button label font color, which changes to the dark gray color used for icons (RGB=99,100,102). There is no change to the button's background color. Here is an example:

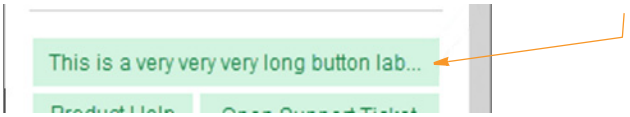


- Once a button is clicked (clicked state), there is no change to the button's appearance.
- The configuration file defines the number of buttons that appear before the More button or includes a setting for each button that determines if the button should appear before or after the More button is clicked.

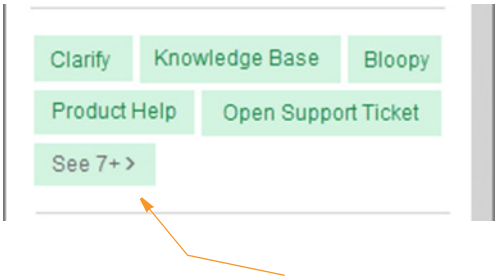
- The maximum width of a button is limited to the width of the gray horizontal separators used in the Agile Desktop UI. Here is an example:

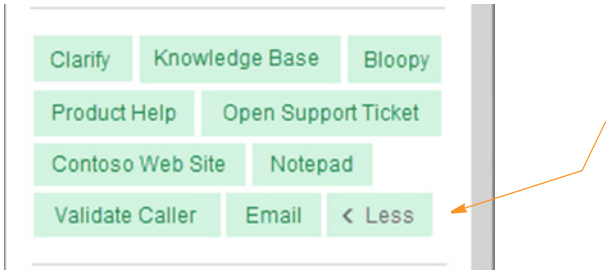


- If a button's label text exceeds the maximum width, shorten the text and append an ellipsis, as shown here:

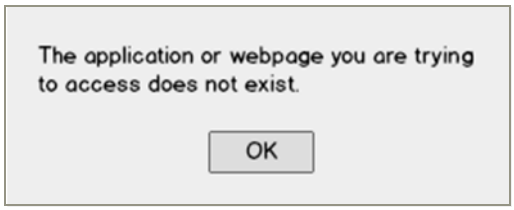


- The More/Less button has the same background color as the other buttons but the label color is the dark gray used for icons, as shown here:





- The More button shows the number of shortcuts that are not displayed.
- Clicking the More button displays all shortcuts and replaces the More button with the Less button.
- If a shortcut fails to launch the associated application or web site, display this message:



- You can have multiple instances of the shortcut plug-in on a single Agile Desktop toolbar.

Adding Links and External Applications

Follow these steps to add a link or an application to the Shortcuts plug-in. Here is an example:

1. Open Interaction.xml file in the Config folder. Scroll to the end of the file and locate this entry:

```
<Plugin name="Shortcuts">
```

```
168         <Plugin name="Shortcuts">
169             <section name="Links">
170                 <Item Name="Validate Caller" Target="ValidateCaller" ShortcutType="Activity"></Item>
171                 <Item Name="OpenSpan Web Site" Target="http://www.openspan.com/" ShortcutType="URL"></Item>
172             </section>
173             <section name="Launch">
174                 <Item Name="Notepad" Target="Notepad" ShortcutType="External"></Item>
175                 <Item Name="Clarify" Target="ClarifyPackage" ShortcutType="OpenSpan"></Item>
176             </section>
177             <section name="Tabs">
178                 <Item Name="CRM" Target="TabName" ShortcutType="Tab" json:Array='true'></Item>
179             </section>
180         </Plugin>
```

2. To add a link to a web site, enter the following:

```
<Item Name="name" Target="URL" ShortcutType="URL"></Item>
```

Parameter	Description
Item Name	Enter the name of the web site. This name will appear on the Application Bar.
Target	Enter the full address of the web site, such as http://www.openspan.com
ShortcutType	For a link, make sure this is set to *URL*.

Here is an example:

```
<Item Name="OpenSpan" Target="http://www.openspan.com"
ShortcutType="URL"></Item>
```

3. To add an application, enter the following inside:

```
<Item Name="name" Target="name and path" ShortcutType="External"></Item>
```

Parameter	Description
Item Name	Enter the name of the application. This name will appear on the Application Bar.
Target	Enter the executable name and its full path, such as C:\Program Files\Microsoft Games\Solitaire\Solitaire.exe
ShortcutType	To launch an external application, make sure this is set to *External*.

Here is an example:

```
<Item Name="Solitaire" Target="C:\Program Files\Microsoft
Games\Solitaire\Solitaire.exe" ShortcutType="External"></Item>
```

4. Save your changes in the Interaction XML file.

Your changes appear the next time you start Agile Desktop.

Using the StartInteraction Plug-In

The StartInteraction plug-in lets you start a search from Agile Desktop. It is typically used to start an interaction. This involves writing an automation that will actually start the interaction, which is discussed in this topic.

Note Keep in mind that you can use this plug-in for any single parameter search.

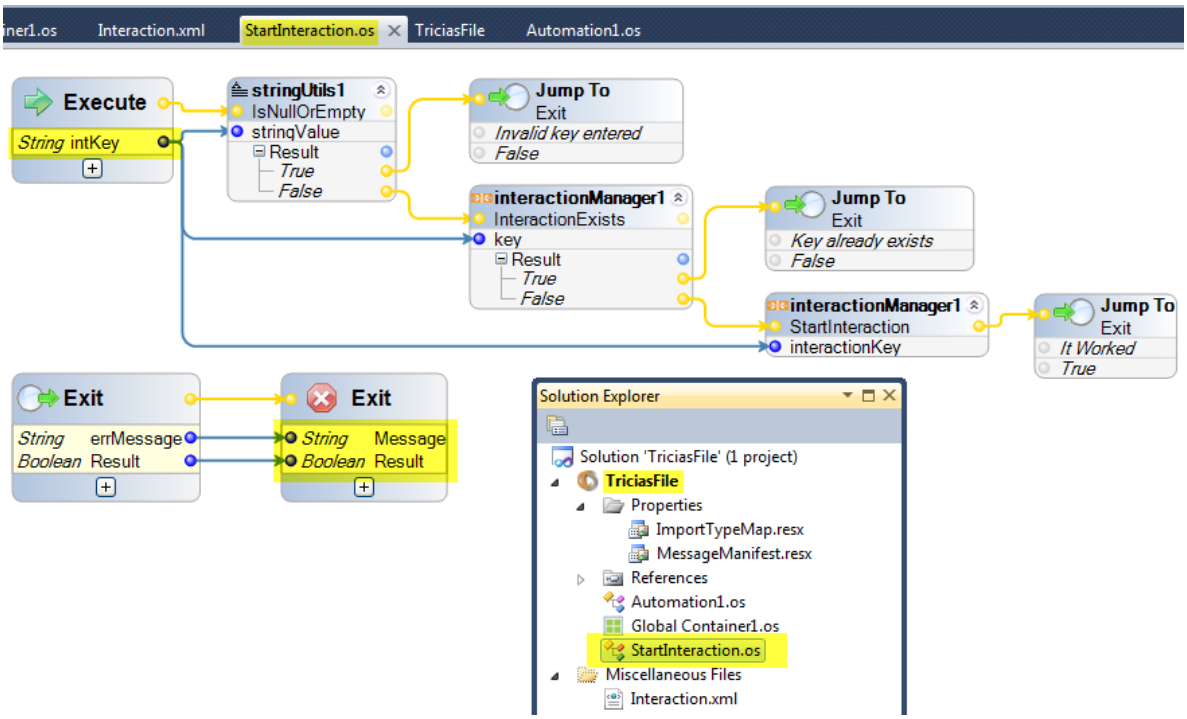
Here is how the plug-in appears in the Interaction.xml file:

```
<Plugin name="StartInteraction" isActive="true" enableWithNoActiveInteraction="true">
    <config>
        <label>Enter Interaction Key</label>
        <project>TriciasFile</project>
        <automation>StartInteraction</automation>
        <defaultErrorMessage>Unable to start interaction</defaultErrorMessage>
    </config>
</Plugin>
```

You define these keys in the Interaction.xml file:

Key	Definition
Label	Defines what appears on the Agile Desktop application bar above the entry field. In this example, it is the text, *Enter Interaction Key*.
Project	Enter the name of the project that contains the automation you want the system to call when the end user clicks the Search icon. This should be a start-up project. Replace any spaces in the project name with underscores. For example, you would change *My Project* to *My_Project*.
Automation	Enter the name of the automation you want the system to call when the end user clicks the Search icon. This should be the name of the project you entered in the Project key. Replace any spaces in the project name with underscores. For example, you would change *My Automation* to *My_Automation*.
defaultError Message	Enter the message you want show if the automation returns False. This message serves as a default and can be overwritten in the automation.

Here is an example automation:



Keep in mind these points:

- The entry point must be able to take a string. This is the string entered by the end user.
- The automation name must match the name entered in the Automation key in the Interaction.xml file — except for changing spaces into underscores.
- The exit point must have a output string. This is the message that appears if the automation returns False.
- The exit point should return a Boolean result. True means the search was successful and no error message was shown. False means the error message should be shown.

Deploying Agile Desktop to End Users

Before you deploy Agile Desktop with your OpenSpan solutions, you need to make sure the system has access to the necessary Agile Desktop files, which are stored in the AgileDesktop folder.

The AgileDesktop folder contains the JavaScript, HTML, CSS, images, and other files required to run Agile Desktop and OpenSpan solutions. This folder is installed with OpenSpan Studio and is automatically configured for Studio, but for an Agile Desktop install, you need to...

- Place the AgileDesktop folder in an accessible location
- Configure each Agile Desktop installation to look for the AgileDesktop folder

Note If you try to run OpenSpan solutions without access to this folder, you will get a fatal error message.

Locating the AgileDesktop Folder

If you installed OpenSpan Studio in the default installation folder, the AgileDesktop folder should exist in one of these paths:

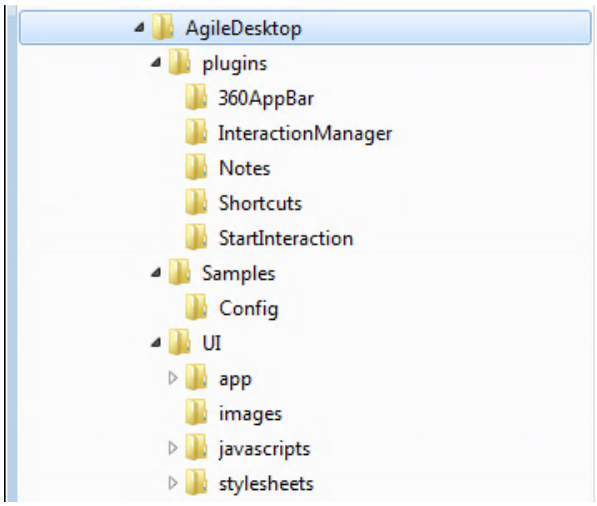
Operating system	Path
64-bit Windows	C:\Program Files (x86)\OpenSpan\OpenSpan Studio for Microsoft Visual Studio 2010\Application\AgileDesktop
32-bit Windows	C:\Program Files\OpenSpan\OpenSpan Studio for Microsoft Visual Studio 2010\Application\AgileDesktop
If you used a custom install path, the folder may exist in a path similar to this: C:\(*custom path*)\OpenSpan\OpenSpan Studio for Microsoft Visual Studio 2010 \Application\AgileDesktop	

For computers which only have Agile Desktop installed — no Studio — you must place the folder in an accessible location and configure Agile Desktop so it can find the folder and the files located in it.

Note The Agile Desktop/Runtime installation does not install the AgileDesktop folder. You must set up access to this folder. This makes it easier to enforce consistency for Agile Desktop users.

Accessing the AgileDesktop Folder

The best way to make sure all Agile Desktop users have access to the same AgileDesktop folder is to host it on an internally available web server. This is the approach OpenSpan recommends. This example shows the contents of the AgileDesktop folder:



Making the AgileDesktop Folder Available via a File Server or Shared Folder

If you have a Network Attached Storage (NAS) or other file sharing server, make the AgileDesktop folder available to your Agile Desktop/Runtime users via that server. Your Agile Desktop/Runtime users should then place the path to the hosted AgileDesktop folder in the Agile Desktop section of their RuntimeConfig.xml files.

This approach means there is a single folder for you to change whenever Agile Desktop is updated for all Agile Desktop/Runtime users.

Making the AgileDesktop Folder Available via Internet Information Services (IIS)

You can also create a virtual folder with a path that leads to the AgileDesktop folder by following the steps in these topics:

For IIS version	See this topic
5.0 and 6.0	How to: Create and Configure Virtual Directories in IIS 5.0 and 6.0
7.0	Create a Virtual Directory (IIS 7.0)

Make sure the virtual folder is accessible by Agile Desktop users and that the permissions set for the virtual folder allow reading. Your Agile Desktop users should then add the Uniform Resource Locator (URL) of the virtual folder to the AgileDesktop section in their RuntimeConfig.xml files. Here is an example:

```
<AgileDesktop>
  <Url value="https://omc.openspan.com/AgileDesktop" />
  <LogoUrl value="http://logo.png" />
  <UserCanHide value="true" />
  <UserCanExit value="true" />
  <TopMost value="true" />
  <ReadUITimeout value="10000" />
  <AccentColor value="Green" />
</AgileDesktop>
```

Once the virtual folder is set up, whenever a hot fix or upgrade is made, you only have to apply the changes in the folder targeted by your Virtual Directory for your entire user base to receive those changes.

Making the AgileDesktop Folder Available on a Single Computer

Note: This method is not recommended since you have to manually upgrade Agile Desktop on each computer when updates are available.

1. Copy the AgileDesktop folder onto the local machine.
2. Open the RuntimeConfig.xml file and change the Url attribute value in the AgileDesktop section to point to the absolute path of the AgileDesktop folder on the local machine.

INDEX

T

U

V

W

X