Sage SalesLogix Migration Tool

Compatible with Sage SalesLogix version 7.5.x and Later

Developed by Sage SalesLogix User Assistance

sage software

Your business in mind.

Sage SalesLogix Migration Tool

Documentation Comments	This documentation was developed by Sage SalesLogix User Assistance. For content revisions, questions, or comments, contact the Sage SalesLogix writers at saleslogix.techpubs@sage.com.
Copyright	Copyright © 1997-2010 Sage Software, Inc. All Rights Reserved. This product and related documentation are protected by copyright and are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sage Software and its licensors, if any.
Version	Compatible with Sage SalesLogix version 7.5.x and Later (072310) 2010
Trademarks	SalesLogix is a registered trademark of Sage Software, Inc. Other product names may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.
Disclaimer	Sage Software has thoroughly reviewed this manual. All statements, technical information, and recommendations in this manual and in any guides or related documents are believed reliable, but the accuracy and completeness thereof are not guaranteed or warranted, and they are not intended to be, nor should they be understood to be, representations or warranties concerning the products described. Sage Software assumes no responsibility or liability for errors or inaccuracies with respect to this publication or usage of information. Further, Sage Software reserves the right to make changes to the information described in this manual at any time without notice and without obligation to notify any person of such changes.

Contents

Introduction 1

Overview 1
About This Guide 1
What You Need to Know 1
Prerequisites 2

Chapter 1 Installing the Migration Tool 3

File Information 3
To install the Migration Tool 3

Chapter 2 Understanding What to Expect from the Migration Tool 5

Preparing your Forms for Migration 5
Migration Results 6
Limitations 6
Upgrade Considerations 7

Chapter 3 Using the Migration Tool 9

Running the Tool 9
Viewing the Migration Report 10
Finishing Conversion 11

Chapter 4 Example: Migrating the Widgets Bundle 13

Installing the Bundle on the Sage SalesLogix Database 13
Migrating the Bundle 14
Locating Migrated Forms 19

Appendix A Reference 21

Control Mapping 21

Introduction

The Migration Tool is a utility that enables you to convert Sage SalesLogix plugins created for the Sage SalesLogix Network Client in Architect into quick forms for use in the Sage SalesLogix Web Client.

Note This version enables migration of forms created in any version of Sage SalesLogix and scripts created in version 6.0 and later.

Overview

Sage SalesLogix is a feature-rich product. Consequently, converting Network Client customizations to the Web Client can be time-intensive. The Migration Tool automates many of the conversion tasks, reducing work by creating forms, scripts, navigation, and data relationships for you.

There are several ways you can use Migration Tool to assist you:

- Use it to make a quick assessment of what customizations will convert.
- Learn about conversion by choosing something simple to migrate and studying the outcome.
- Use it for a full migration of your customizations.

Porting a Sage SalesLogix Network Client customization can be divided into several phases:

- Use the Architect to make recommended simplifications to your forms and create a project.
- Use the Application Architect to run the Migration Tool on the legacy project. The Migration Tool converts bound ActiveX forms and scripts to SalesLogix metadata including quick forms definitions, logs conversions, and separates strings and business logic into .NET assemblies.
- If necessary, adjust the legacy forms and rerun the Migration Tool to fine-tune the migration results.
- Review newly converted items and adjust Web forms and business logic. Features of the Application Architect such as Remove Row/Column and Cut/Copy/Paste on quick forms, and the Data Sources window can be used to refine your Web form layout.

About This Guide

This guide provides information for the technological preview of the Migration Tool utility. This tool is in limited release, and both utility and document are subject to change.

If you have comments about this document, please send them to saleslogix.techpubs@sage.com.

What You Need to Know

This guide assumes that you have a working knowledge of the Architect and the Application Architect. The purpose of this guide is to provide information to assist you as you migrate Sage SalesLogix Network Client customizations to Web Client implementations. This guide is not designed to teach you to customize Sage SalesLogix.

Prerequisites

This version of the Migration Tool requires Sage SalesLogix version 7.5.x.

Chapter 1
Installing the Migration Tool

To install the Migration Tool, extract files from the Migration Tool zip file and edit two files.

File Information

The Migration Tool zip file contains the following files:

Migration Files
Migration\Borland.Vcl.dll
Migration\Interop.AxSLXCharts.dll
Migration\Interop.AxSLXControls.dll
Migration\Interop.AxSLXDialogs.dll
Migration\Interop.SalesLogix.dll
Migration\Interop.SLXCharts.dll
Migration\Interop.SLXControls.dll
Migration\Interop.SLXDialogs.dll
Migration\Interop.SLXOptions.dll
Migration\Interop.StdType.dll
Migration\Interop.StdVCL.dll
Migration\Sage.SalesLogix.Migration.dll
Migration\Sage.SalesLogix.Migration.Forms.dll
Migration\Sage.SalesLogix.Migration.Module.dll
Migration\Sage.SalesLogix.Migration.Script.dll
Sample\Widgets.sxb
Sage SalesLogix Migration Tool Guide.pdf

To install the Migration Tool

1. Ensure that Sage SalesLogix 7.5.x is installed.
2. Extract the files and folder structure from SLX_v75x_MigrationTool.zip, to ..\Program Files\SalesLogix.
3. In a text editor, open the file named ..\Program Files\SalesLogix\SageAppArchitect.exe.config.
4. Locate the line containing `<probing privatePath="Platform;SupportFiles"/>` and replace it with `<probing privatePath="Platform;SupportFiles;Migration"/>`
5. In a text editor, open the file named ..\Program Files\SalesLogix\AppConfig\SalesLogix.xml.

   Caution Incorrect information in this file can prevent the Application Architect from opening.
6. Add the following new <Include> child to the <Modules> section beneath the existing children:
   ```
   <Include ModuleName=
   "Sage.SalesLogix.Migration.Module.MigrationModule,Sage.SalesLogix.Migration.Module"/>
   ```
7. Restart the Application Architect, and then confirm that the toolbar contains the Migration Tool button.

Chapter 2
Understanding What to Expect from the Migration Tool

Consider the following plugin elements created using Architect as you plan your conversion to the Web Client:

- Interface - contains controls and layout of the plugin.
- Data binding - provides a way to create a read/write link between the controls on a form and the data in the data model.
- Scripting - enables the form to function and defines or constrains the workflow as it applies to this particular plugin.

When you use the Migration Tool to convert plugins into quick forms, you can expect the following:

- Elements in the user interface convert.
- Data binding converts.
- Scripting converts, but as unmanaged code not associated to the form. After migration, you will need to locate the converted code in the .cs or .vb project files, add it to your assembly, create business rule definitions, and link the events to the business rules.

Note This impacts any situation where functionality is achieved through scripting and data is programmatically fetched and bound to forms.

Preparing your Forms for Migration

The Migration Tool targets conversions to benefit the most users; consequently it completes some tasks more effectively than others. For example, forms that have been created using basic best practice conventions convert more successfully than forms that have not. You can improve the success of your migration by making adjustments to your plugins before using the Migration Tool. Alternatively, you can choose to do minimal preparation using the following suggestions and make adjustments to the converted forms.

- Remove overlapping controls. This includes hidden controls, non-visual controls, and shapes.
- Give labels consistent widths within their "columns", especially when they are visually right-aligned to controls. Do not autosize.
- Simplify overly complex layouts. For example, limit the number of panels in a form, and avoid putting panels inside panels.
- Ensure each form is bound to the appropriate entity.

Migration Results

After migrating your plugins to quick forms, you can expect to see the following results:

- The migration creates entities under Entity Model>Packages>(specified package).
- Bi-directional joins stored as pairs of SQL joins are converted to one-to-many with an include.
- Flat buttons become labels.
- Image buttons become buttons.
- Any label to the immediate left of a control (i.e., with no intervening control) is incorporated into the control as its Caption property. All other labels are treated as separate controls.
- Code behind forms converts to the .NET language you specified, but is not automatically attached to the form. It is stored as unmanaged code at the location you entered in Output Directory field. You will need to open the files, review and change as appropriate, and then manually attach the code to the form.
- Forms are placed in the Entity Model under the entities to which they are bound. For example, if the form is bound to the Account table, the form converts under Account. Unbound forms (for example, Managed forms) are placed in the Entity Model under the entity you selected in the Main Table field.
- When the migration encounters a control type that is not supported in the Application Architect, the error log displays errors in red text and warnings in purple.
- An error is generated when a binding string cannot be converted into a property string.
- All converted scripts are stored in one .NET directory.
- Bound controls that are set to invisible are placed in a hidden object control container at the bottom of the form.
- Controls that overlap are logged in the migration output and converted.
- A best-fit layout is used for the converted forms. You can use the Application Architect to quickly fine-tune your layouts. See the section entitled “Finishing Conversion” on page 11 for suggestions on how to adjust your layouts.

Limitations

The following do not convert in this version of the Migration Tool.

- Combo boxes do not show a date format when converted.
- Functionality achieved through scripting is not migrated into the new quick form metadata.
- Non-visual controls such as font/file/color dialogs, timers, and popup menus have no equivalent in quick forms, so they do not convert.
- Contract fields are not formatted as currency.
- Security for the Web platform is on a per user basis, which is not structured for team-based security.
- Data binding on grids and lookups is limited to single level 1:M joins. If the query behind the legacy control contains multiple levels of 1:M joins, the joins beyond the first level are excluded. For example, if you have a grid on an account tab view that displays all addresses of all contacts (account --< contact --< address), the address information is omitted. Standard attachment grids are affected in the same way. Because they have a second 1:M join (to a table for Remote Client information), the column with the invalid join is excluded. The Migration Report error message says, “Error: Invalid join direction in ’ATTACHMENT.ATTACHID=ATTACHID.REMOTEATACHMENTS’ join string”.
- Some SalesLogix ActiveX and Enable Basic controls do not convert. See the tables beginning on page 21 for a complete list.

Upgrade Considerations

You can migrate your Sage SalesLogix Network Client customizations created in Sage SalesLogix v6.x or later directly to a Web implementation using Sage SalesLogix 7.5.x. You do not need to take action on your customizations prior to upgrading to 7.5.x. To migrate your customizations after upgrading you would complete the following:

1. Ensure you have the 7.5.x schema (installed via the Upgrade bundle).
2. If necessary, in Architect, create a new project that contains your custom plugins and associated information.

   If you created a project or bundle of your customizations prior to upgrading to 7.5.x, you do not need to create a new project.
3. Migrate the 6.x or later project to the 7.5.x Web Client using the Migration Tool.
4. Do a side-by-side comparison to plan the work to reintegrate your customizations.

Chapter 3
Using the Migration Tool

Running the Tool

After you have installed the Migration Tool and prepared your forms, you are ready to begin your migration.

To migrate

1. Using Application Architect, create a project and add the plugins you want to migrate.

   Recommended practice is to package forms and data together. The Migration Tool uses the entity information as it converts the forms.

   For more information on creating projects, see the Architect Help topic called “Creating and Saving Projects.”
2. In the Application Architect toolbar, click the **Migration Tool** button.

   The Migration Tool dialog box opens.
3. In the **Legacy Project** drop-down list box, select the project you created in step 1.

   If the project is not visible, verify that the bundle is installed in the Sage SalesLogix database, and then restart the Application Architect.
4. In the **Target Package** drop-down list box, type a new name or select the package you want to contain the migrated entities. The default is Sage SalesLogix Application Entities.

   A package within Entity Explorer is a development container in which customizations can be managed for deployment to the Web Server.

   If you specify a new package, connections to the portal will be created for you.

   Migrated entities are placed in the target package unless they refer to an existing table, for example ACCOUNT. In this case, the migrated entity is placed in the same package as the existing table.

   Migrated entities become visible in the Project Workspace (as “VFS on (project name)”) when you expand Entity Model > Packages. For more information on packages and entities, see the Application Architect Help topic called “Managing Entities.”
5. In the **Target Portal** drop-down list box, select the portal you want to host your migrated bundle. The default is Sage SalesLogix Client.

   A portal represents a collection of Web pages. This is where the smart part mappings, navigation, and menus will go. For more information on portals, see the Application Architect Help topic called “Managing Portals.”
6. In the **Target Manifest** drop-down list box, type a new name or select an existing bundle manifest to contain your migrated forms.

   The manifest contains a transportable list of everything that is migrated. It will contain any new entity that does not map to an existing table in the database

   Packages not flagged to participate in the platform build are excluded from the manifest. For more information on platform builds, see the Application Architect Help topic called “Building the Web Platform”.
7. In the **Main Table** drop-down list box, select the entity under which forms not bound to data will be inserted. The default is ACCOUNT.
8. In the **Namespace** box, type the prefix for your new assembly.

   This should be unique and without spaces because this will be used as a coding namespace. For example, SalesLogix.Widget.

9. In the **Output Directory** drop-down list box, select the location where migrated source code will be stored.

   A Visual Studio project file is created at this location.
10. Select the **Process Scripts** check box to migrate scripts in your forms. Clearing this check box removes this step from the migration and disables the VS Project Name and Target Language options. You might bypass script generation if the project has no scripts or you are focusing on the layout of your forms.
11. If necessary, in the **VS Project Name** box, type the file name for the Visual Studio project file.

    This file will contain references to all the generated source code files and a file for each VBScript (including script includes), nested in a directory named for the VBScript family. Interop wrappers are generated for libraries referenced by CreateObject in code or ActiveX controls on forms and placed at the root of the directory you entered in step 9 of the migration process.
12. If necessary, in the **Target Language** section, select the .NET programming language to use in your migrated forms.

    If you select the Custom option, you can use any third party .NET language.
13. In the **Strong Name Key Pair** box, specify or browse to the key for strong naming automatically generated interop wrappers.

    Use this field if your forms contain custom ActiveX controls or your script uses custom COM libraries.
14. If you want label and control pairs to convert as separate controls, clear **Merge labels with adjacent controls**.
15. Click **Run**.

    Migration converts the forms and displays messages in the Output window (> Show output from: Migration Output.) Errors appear in red and warnings display in purple. The Migration Output persists until you run another migration or build a new interface.
16. Click **Close**.
17. Pre-view your form layout by opening the form in the Application Architect.
18. To view it in the Web Client, build the Web platform, and deploy and open the portal.

    See the Application Architect Help topics “Building the Web Platform” and “Deploying the Sage SalesLogix Web Client Portal” for more information.

    You are now ready to view the Migration Report, rerun the Migration Tool, or finalize the layout of your converted form.

Tip You can rerun the Migration Tool repeatedly until you have made all of your pre-migration adjustments. After the last run, use the Application Architect to make your post-Migration adjustments.

Viewing the Migration Report

The migration generates a report summarizing the migration results for each plugin grouped by plugin type. The end of the report shows a record of the data you entered in the Migration Tool dialog box.

To view the migration report

1. Ensure the Migration Tool dialog box is closed.
2. In the Application Architect, on the **View** menu, click **Migration Reports**.

   A list of Migration Reports by run date and time displays as a column of buttons in the tabbed MDI window next to the Project Explorer window.

3. Click the button of the report you want to open.

   The report opens in another tab. The label on the tab includes the project name and the migration run date and time.

4. If necessary, open entities listed in the Migration Report by clicking their names.

   Entity names displayed in red open in the quick forms designer at the location of the error.

Finishing Conversion

Once you are finished running the Migration Tool on your package, open the forms, correct any issues in the interface, and then add the scripting and business rules.

To finish converting your forms

1. Ensure you have corrected any issues visible in the **Migration Report**.
2. Open each of your migrated forms and correct any formatting issues.
   a. In the **Project Workspace Manager**, click the default VFS project workspace.
   b. Expand **Entity Model**, expand **Packages**, and click the package you specified in step 5 of the migration process.

      A list of entities in the package displays.

   c. To expand the list of forms bound to an entity, click the entity name and then **Forms** below it. For forms not bound to data, use the entity name you specified in step 7 of the migration process.

      The list of forms that displays will include the forms that were created during migration.

   d. Click a form to open it in the Quick Forms Editor.
   e. Review the form and make any necessary edits.

      At first the converted form may seem too complex or even unrecognizable. You can easily correct the formatting with a few quick clicks:

      - Right-click to select an unoccupied column or row (appears very narrow), and choose Remove Column or Remove Row. Repeat until a message appears indicating the column or row is not empty. Controls align as you delete empty columns and rows.
      - Change any control that seems too wide or tall by reducing its ColumnSpan or RowSpan property to 1 or 2.
      - Move controls using the right-click Cut/Copy/Paste functionality.

      Keep in mind that the Quick Form editor does not show an exact representation of the form as it will look in the Web Client; before you make extensive changes to the form, you may want to review its rendering in the Web Client. You will need to build the Web platform and deploy the portal. See the Application Architect Help topics “Building the Web Platform” and “Deploying the Sage SalesLogix Web Client Portal” for more information.

3. Add scripting.

   Migrated scripts are stored in the location you entered in step 9 of the migration process. If you opted to migrate existing scripts, review the migrated scripts carefully to determine if they are still required in the new environment and in what ways the content needs to be edited. Then connect the migrated scripts to the new forms.

Chapter 4
Example: Migrating the Widgets Bundle

This example provides the steps for migrating the Widgets bundle from Sage SalesLogix ActiveX forms into .aspx forms using the Migration Tool utility in the Application Architect. The example assumes you have successfully installed the Migration Tool utility in a Sage SalesLogix 7.5.x environment.

The Widgets bundle contains the following plugins:

Widget Plugin	Type
Account Widget	Form for the Tab
Account Widget Add-Edit	Form
Widget Add-Edit	Form
Widget Details	Main View
Widget Details	Form
Widget Menu	Menu
Widget Toolbar	Toolbar
Manage Widgets	Form
C_Account_Widgets	Table
Widgets	Table
Widget	Lookup
Bevel And MultiTab Form	Form
Widget Library	Scripts, VBscript

Installing the Bundle on the Sage SalesLogix Database

First, install the Widgets.sxb on your Sage SalesLogix database.

To install

1. Confirm that Widgets.sxb was extracted from the Migration Tool zip file to ..\Program Files\SalesLogix\Sample.
2. Open the Administrator and install the Widgets.sxb bundle.
    a. On the **Bundle Manager** toolbar, click **Install**.
    b. In the **Open** dialog box, browse to the Widgets.sxb and click **Open**.
    c. In the **Choose Actions to Install** dialog box, select all of the plugins and then click **OK**.
    d. Click **Yes** to replace plugins in the database.
    e. Click **Yes** to release the plugins.
    f. Select the team(s) to which the plugins are released, and click **OK**.

    You are now ready to migrate the bundle.

    For more information about installing plugins, see the “Installing a Bundle” topic in the Administrator online Help.

Migrating the Bundle

For the purpose of this example, we will deploy the migrated Widget customization to the Sage SalesLogix Client Web site. The following are default settings for the Sage SalesLogix Client Web site.

Alias: SlxClient

Title: Sage SalesLogix

Description: Sage SalesLogix

Template URL: Masters\base.master

Server: localhost

Deploy Path: c:\inetpub\wwwroot\SlxClient

Virtual Directory name: SlxClient

To migrate

1. From the Application Architect toolbar, click the **Migration Tool** button.

   The Migration Tool dialog box opens.
2. In the **Legacy Project** drop-down list box, select **Widgets**.
3. In the **Target Package** drop-down list box, select **Sage Application Entities**.
4. In the **Target Portal** drop-down list box, select **Sage SalesLogix**.
5. In the **Target Manifest** drop-down list box, type **Widgets**.
6. In the **Main Table** drop-down list box, select one of the available tables, for example, **ACCOUNT**.
7. In the **Namespace** box, type SalesLogix.Widgets.
8. In the **Output Directory** box, type C:\Test.

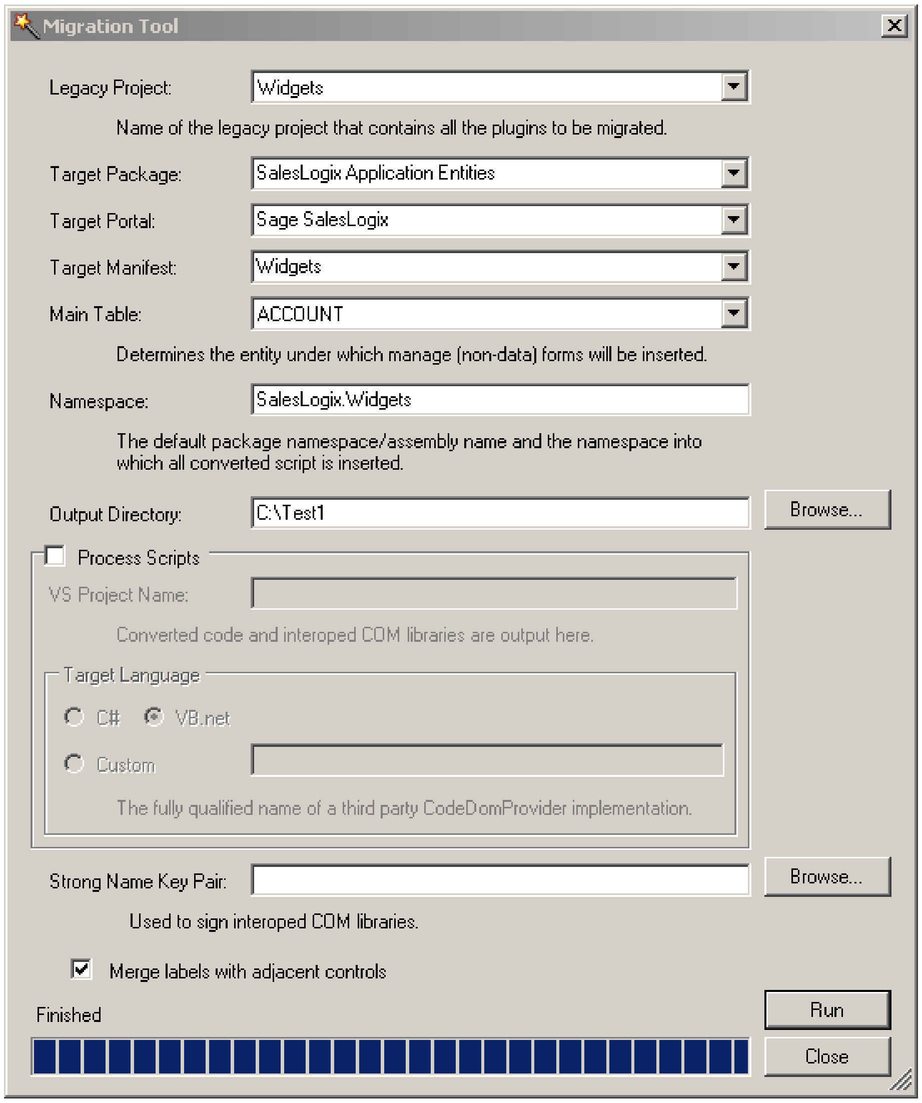
   Migrated source code will be stored under C:\Test.
9. Clear the **Process Scripts** check box, because there are no scripts in this example bundle.

   This disables the VS Project Name and the Target Language boxes.
10. Leave the **Strong Name Key Pair** box blank.

    This is the location you would save to if your forms use a third-party product such as Excel or CTI. The Widgets bundle does not.
11. Check **Merge labels with adjacent controls**.

    This causes the labels to be used for the Caption property on adjacent controls.

12. Click **Run**.



Migration Tool

Legacy Project: Widgets

Name of the legacy project that contains all the plugins to be migrated.

Target Package: SalesLogix Application Entities

Target Portal: Sage SalesLogix

Target Manifest: Widgets

Main Table: ACCOUNT

Determines the entity under which manage (non-data) forms will be inserted.

Namespace: SalesLogix.Widgets

The default package namespace/assembly name and the namespace into which all converted script is inserted.

Output Directory: C:\Test1

☐ Process Scripts

VS Project Name:

Converted code and interoped COM libraries are output here.

Target Language: C# [✓ VB.net] Custom

The fully qualified name of a third party CodeDomProvider implementation.

Strong Name Key Pair:

Used to sign interoped COM libraries.

☑ Merge labels with adjacent controls

Finished

Migration displays messages in the Output window and converts the forms. When migration is done, the status message will say Finished or Error.

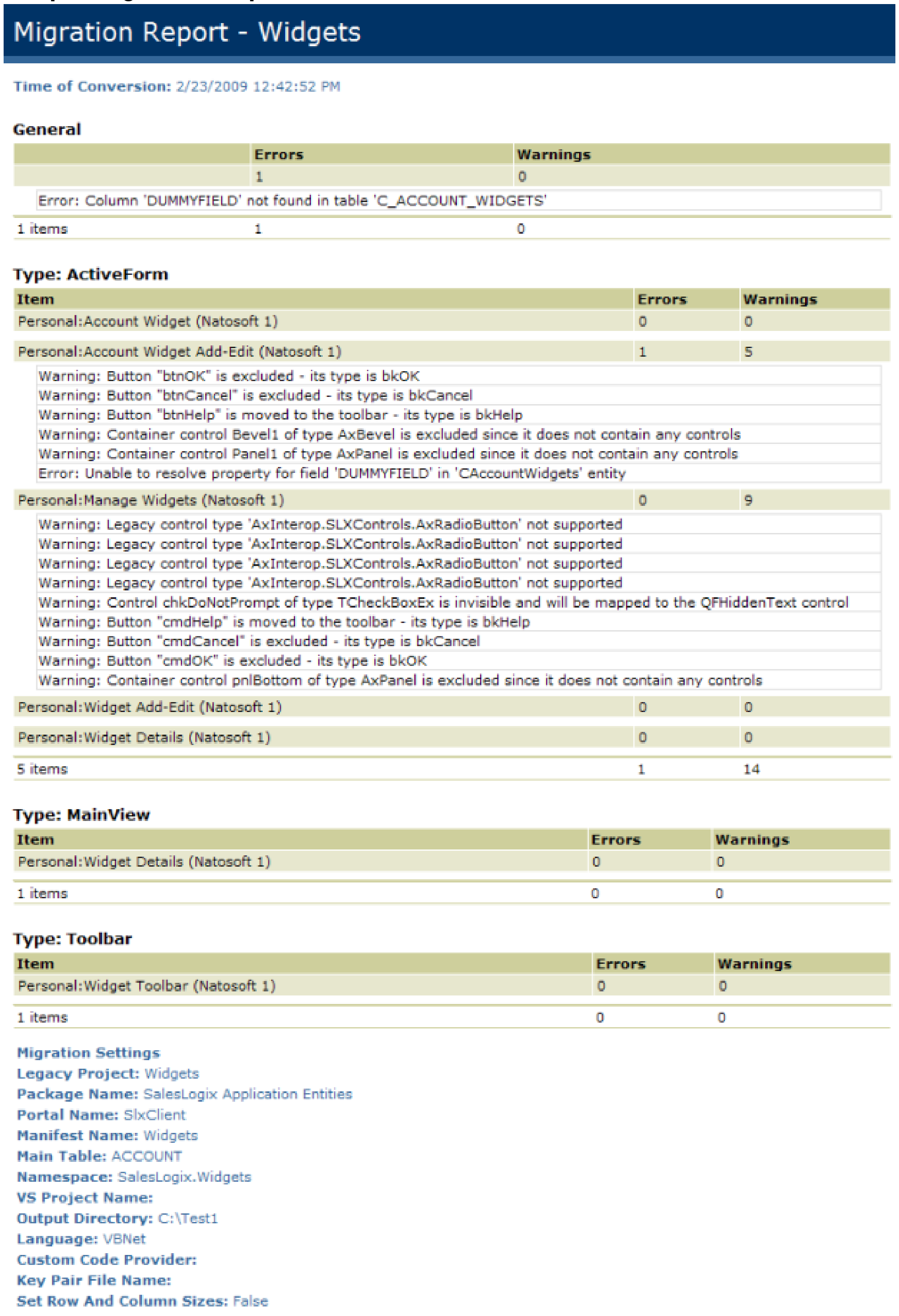
13. Click **Close** and view the Output Window.

Example Output Window:

```
INFO  - Parsing '[MainView] Personal:Widget Details (SalesLogix 1)' main view
INFO  - Parsing '[Toolbar] Personal:Widget Toolbar (SalesLogix 1)' toolbar
INFO  - Parsing '[ActiveForm] Personal:Bevel And MultiTab Form (SalesLogix 1)' form
WARN  - [ActiveForm] Personal:Bevel And MultiTab Form (SalesLogix 1) - Button "Button1" is excluded - its type is bkOK
WARN  - [ActiveForm] Personal:Bevel And MultiTab Form (SalesLogix 1) - Button "Button2" is excluded - its type is
bkCancel
WARN  - [ActiveForm] Personal:Bevel And MultiTab Form (SalesLogix 1) - Button "Button3" is moved to the toolbar - its
type is bkHelp
INFO  - Parsing '[ActiveForm] Personal:Manage Widgets (SalesLogix 1)' form
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Legacy control type 'AxInterop.SLXControls.AxRadioButton'
not supported
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Legacy control type 'AxInterop.SLXControls.AxRadioButton'
not supported
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Legacy control type 'AxInterop.SLXControls.AxRadioButton'
not supported
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Legacy control type 'AxInterop.SLXControls.AxRadioButton'
not supported
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Control chkDoNotPrompt of type TCheckBoxEx is invisible
and will be mapped to the QFHiddenText control
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Grid grdTargets is excluded since it is not bound
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Button "cmdHelp" is moved to the toolbar - its type is
bkHelp
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Button "cmdCancel" is excluded - its type is bkCancel
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Button "cmdOK" is excluded - its type is bkOK
INFO  - Parsing '[ActiveForm] Personal:Widget Details (SalesLogix 1)' form
INFO  - Parsing '[ActiveForm] Personal:Account Widget (SalesLogix 1)' form
INFO  - Parsing '[ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1)' form
WARN  - [ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1) - Button "btnOK" is excluded - its type is bkOK
WARN  - [ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1) - Button "btnCancel" is excluded - its type is
bkCancel
WARN  - [ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1) - Button "btnHelp" is moved to the toolbar - its
type is bkHelp
INFO  - Parsing '[ActiveForm] Personal:Widget Add-Edit (SalesLogix 1)' form
INFO  - Resolving 'WIDGET' table
INFO  - Resolving 'ACCOUNT' table
INFO  - Resolving 'LEADSOURCE' table
INFO  - Resolving 'PRODUCT' table
INFO  - Resolving 'C_ACCOUNT_WIDGETS' table
INFO  - Resolving 'WIDGET' table relationships
INFO  - Resolving 'ACCOUNT' table relationships
INFO  - Resolving 'LEADSOURCE' table relationships
INFO  - Resolving 'PRODUCT' table relationships
INFO  - Resolving 'C_ACCOUNT_WIDGETS' table relationships
INFO  - Building 'Personal_Widget Details' main view
INFO  - Building 'Personal_Widget Toolbar' navigation
INFO  - Building 'Personal_Bevel And MultiTab Form' form
WARN  - [ActiveForm] Personal:Bevel And MultiTab Form (SalesLogix 1) - Container control Bevel1 of type AxBevel is
excluded since it does not contain any controls
WARN  - [ActiveForm] Personal:Bevel And MultiTab Form (SalesLogix 1) - Container control Panel1 of type AxPanel is
excluded since it does not contain any controls
INFO  - Building 'Personal_Manage Widgets' form
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Container control pnlBottom of type AxPanel is excluded
since it does not contain any controls
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Control 'lblCreateDateTo' overlaps control
'dteCreateDateTo'
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Control 'lblInclude' overlaps control 'lblSelect'
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Control 'lblTargets' overlaps control 'cmdAddFromGroup'
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Control 'lblTargets' overlaps control 'cmdAddTargets'
WARN  - [ActiveForm] Personal:Manage Widgets (SalesLogix 1) - Control 'lblTargets' overlaps control 'cmdRemove'
INFO  - Building 'Personal_Widget Details' form
INFO  - Building 'Personal_Account Widget' form
INFO  - Building 'Personal_Account Widget Add-Edit' form
WARN  - [ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1) - Container control Bevel1 of type AxBevel is
excluded since it does not contain any controls
WARN  - [ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1) - Container control Panel1 of type AxPanel is
excluded since it does not contain any controls
ERROR - [ActiveForm] Personal:Account Widget Add-Edit (SalesLogix 1) - Unable to resolve property for field 'DUMMYFIELD
in 'CAccountWidgets' entity
INFO  - Building 'Personal_Widget Add-Edit' form
INFO  - Post building 'Personal_Bevel And MultiTab Form' form
INFO  - Post building 'Personal_Manage Widgets' form
INFO  - Post building 'Personal_Widget Details' form
INFO  - Post building 'Personal_Account Widget' form
INFO  - Post building 'Personal_Account Widget Add-Edit' form
INFO  - Post building 'Personal_Widget Add-Edit' form
INFO  - Persisting 'Personal_Widget Details' main view
INFO  - Item successfully migrated
INFO  - Persisting 'Personal_Widget Toolbar' navigation
INFO  - Item successfully migrated
INFO  - Persisting 'Personal_Bevel And MultiTab Form' form
INFO  - Item successfully migrated
INFO  - Persisting 'Personal_Manage Widgets' form
INFO  - Item successfully migrated
INFO  - Persisting 'Personal Account Widget' form
INFO  - Item successfully migrated
INFO  - Persisting 'Personal_Account Widget Add-Edit' form
INFO  - Item successfully migrated
INFO  - Bundling 'Personal_Widget_Add_Edit' smart part mapping
INFO  - Bundling 'Personal_Bevel_And_MultiTab_Form' smart part mapping
INFO  - Bundling 'Personal_Widget_Details' smart part mapping
INFO  - Bundling 'Personal_Account_Widget' smart part mapping
INFO  - Bundling 'Personal_Account_Widget_Add_Edit' smart part mapping
INFO  - Bundling 'Personal_Widget Toolbar' navigation
INFO  - Finished
INFO  - Persisting 'Personal_Widget Add-Edit' form
INFO  - Item successfully migrated
INFO  - Bundling 'Widget' entity
INFO  - Bundling 'Account' entity
INFO  - Bundling 'Lead Source' entity
INFO  - Bundling 'Product' entity
INFO  - Bundling 'C Account Widgets' entity
INFO  - Bundling 'Personal_Bevel And MultiTab Form' form
INFO  - Bundling 'Personal_Manage Widgets' form
INFO  - Bundling 'Personal_Widget Details' form
INFO  - Bundling 'Personal_Account Widget' form
INFO  - Bundling 'Personal_Account Widget Add-Edit' form
INFO  - Bundling 'Personal_Widget Add-Edit' form
INFO  - Bundling 'Sage.SalesLogix.Migration.RelationshipInfo' relationship
INFO  - Bundling 'Sage.SalesLogix.Migration.RelationshipInfo' relationship
INFO  - Bundling 'Sage.SalesLogix.Migration.RelationshipInfo' relationship
INFO  - Bundling 'SupportFiles\Bin\Sage.SalesLogix.Entities.dll' linked file
```

14. View the Migration Report in the MDI window.

Example Migration Report



Migration Report - Widgets

Time of Conversion: 2/23/2009 12:42:52 PM

General

	Errors	Warnings
	1	0
Error: Column 'DUMMYFIELD' not found in table 'C_ACCOUNT_WIDGETS'		
1 items	1	0

Type: ActiveForm

Item	Errors	Warnings
Personal:Account Widget (Natosoft 1)	0	0
Personal:Account Widget Add-Edit (Natosoft 1)	1	5
Warning: Button "btnOK" is excluded - its type is bkOK		
Warning: Button "btnCancel" is excluded - its type is bkCancel		
Warning: Button "btnHelp" is moved to the toolbar - its type is bkHelp		
Warning: Container control Bevel1 of type AxBevel is excluded since it does not contain any controls		
Warning: Container control Panel1 of type AxPanel is excluded since it does not contain any controls		
Error: Unable to resolve property for field 'DUMMYFIELD' in 'CAccountWidgets' entity		
Personal:Manage Widgets (Natosoft 1)	0	9
Warning: Legacy control type 'AxInterop.SLXControls.AxRadioButton' not supported		
Warning: Legacy control type 'AxInterop.SLXControls.AxRadioButton' not supported		
Warning: Legacy control type 'AxInterop.SLXControls.AxRadioButton' not supported		
Warning: Legacy control type 'AxInterop.SLXControls.AxRadioButton' not supported		
Warning: Control chkDoNotPrompt of type TCheckBoxEx is invisible and will be mapped to the QFHiddenText control		
Warning: Button "cmdHelp" is moved to the toolbar - its type is bkHelp		
Warning: Button "cmdCancel" is excluded - its type is bkCancel		
Warning: Button "cmdOK" is excluded - its type is bkOK		
Warning: Container control pnlBottom of type AxPanel is excluded since it does not contain any controls		
Personal:Widget Add-Edit (Natosoft 1)	0	0
Personal:Widget Details (Natosoft 1)	0	0
5 items	1	14

Type: MainView

Item	Errors	Warnings
Personal:Widget Details (Natosoft 1)	0	0
1 items	0	0

Type: Toolbar

Item	Errors	Warnings
Personal:Widget Toolbar (Natosoft 1)	0	0
1 items	0	0

Migration Settings
Legacy Project: Widgets
Package Name: SalesLogix Application Entities
Portal Name: SlxClient
Manifest Name: Widgets
Main Table: ACCOUNT
Namespace: SalesLogix.Widgets
VS Project Name:
Output Directory: C:\Test1
Language: VBNet
Custom Code Provider:
Key Pair File Name:
Set Row And Column Sizes: False

15. Correct the DUMMYFIELD demonstration error and rerun Migration Tool.

16. To view the migrated forms in the Web Client, build the Web platform and deploy and open the portal. For more information, refer to the Application Architect online help topics “Building the Web Platform” and “Deploying a Portal”.

Locating Migrated Forms

The following table shows where you will find your migrated forms:

Plugin Name	Location of migrated form
Account Widget (Form for the Tab)	Entity Model>Packages>Widgets>AccountsWidgets>Forms
Account Widget Add-Edit (Form)	Entity Model>Packages>Widgets>Widget>Child Properties
Widget Add-Edit (Form)	Entity Model>Packages>Widgets>Widget>Forms
Widget Details (Main View)	Portal>Sage SalesLogix>Pages>Personal_Widget Details
Widget Details (Form)	Entity Model>Packages>Widgets>Widget>Forms
Bevel And MultiTab Form (Form)	Entity Model>Packages>Widgets>Widget>Forms
Widget Toolbar (Toolbar)	
Manage Widgets (Form)	
Widget Menu (Menu)	
C_Account_Widgets (Table)	Not migrated. Already added to the database when the bundle was added to the Administrator.
Widgets (Table)	Not migrated. Already added to the database when the bundle was added to the Administrator.
Widget (Lookup)	Functionality incorporated into Lookup controls.
Widget Library (Scripts, VBscript)	Specified output directory

Appendix A
Reference

This appendix provides information about control mapping.

Control Mapping

There are three types of controls in Sage SalesLogix:

- Sage SalesLogix Quick Forms controls (added in version 7.2)
- Sage SalesLogix ActiveX controls (added in version 6.0)
- Sage SalesLogix Enable Basic controls

The following table shows how the Migration Tool maps Sage SalesLogix ActiveX controls to Sage SalesLogix Quick Forms controls. The Migration Report warns of any Sage SalesLogix ActiveX controls that have no equivalent Sage SalesLogix Quick Forms control.

Sage SalesLogix ActiveX controls	Sage SalesLogix Quick Forms control
Animate	
AutoComplete	
Bevel	Panel (Style Scheme = Bevel)
Browser	Web Browser
Button	Button
Check Box	CheckBox
CheckListBox	
Combo Box	ComboBox
DataGrid	Data Grid
DateTimeEdit	DateTimePicker
DateTimePicker	DateTimePicker
DragSource	
Edit	TextBox
Edit FormatType = ftOwner	Owner
Edit FormatType = ftUser	User
GroupBox	Panel
Image	Image
Label	Label
LinkEdit	TextBox
LinkEdit LinkMode = lemEMail	Email
LinkEdit LinkMode = lemPager	Phone
LinkEdit LinkMode = lemPhone	Phone
LinkEdit LinkMode = lemWebLink	URL

Sage SalesLogix ActiveX controls	Sage SalesLogix Quick Forms control
ListBox	ComboBox
ListView	
LookupEdit	Lookup
LookupEdit LookupMode = lmOwner	Owner
LookupEdit LookupMode = lmUser	User
Memo	Textbox
MonthCalendar	
NameEdit	Person Name
Panel	Panel
PickList	PickList
PickListCombo	
PopupEdit ButtonType = ptEmail	Email
PopupEdit ButtonType = ptPager	Phone
PopupEdit ButtonType = ptPhone	Phone
PopupEdit ButtonType = ptSmallGlobe	URL
PopupMenu	
ProgressBar	
RadioButton	
RadioGroup	Radio Group
RichEdit	
ScrollBar	
Shape	
SOAPClient	
SplitterPanel	
TabControl	MultiTab
Timer	
TrackBar	
TreeView	
UpDown	
SQLLink	
TrackBar	
TreeView	

Table 1: Mapping Sage SalesLogix ActiveX Controls to Sage SalesLogix Quick Form Controls

The following table shows how the Migration Tool maps Sage SalesLogix Enable Basic controls to Quick Forms controls. The Migration Report warns when controls have no equivalent Sage SalesLogix Quick Forms control.

Sage SalesLogix Enable Basic control	Sage SalesLogix Quick Forms control
Bevel	
WebBrowser	Web Browser
Button	Button
Check Box	CheckBox
Combo Box	ComboBox
Data Grid	Data Grid
DateEdit	DateTimePicker
EditBox	TextBox
EditBox FormatType = ftOwner	Owner
EditBox FormatType = ftUser	User
Image	Image
Label	Label
LabelButton	Button
LinkEdit	Email
LinkEdit LinkMode = lemEMail	Email
LinkEdit LinkMode = lemOwner	Owner
LinkEdit LinkMode = lemPager	Phone
LinkEdit LinkMode = lemPhone	Phone
LinkEdit LinkMode = lemUser	User
LinkEdit LinkMode = lemWebLink	URL
LinkLabel	
ListBox	ComboBox
Memo	Person Name
Panel	
PickList	PickList
ProgressBox	
Radio Button	
RadioGroup	Radio Group
ScrollBar	
Shape	
Splitter	
SQLLink	
TrackBar	
TreeView	

Table 2: Mapping Sage SalesLogix Enable Basic Controls to Quick Forms Controls