Portrait
Foundation

Software Development Kit User Guide

Edition 4.0

30 June 2014

Pitney Bowes
Software

Portrait Foundation
Software Development Kit User Guide

Acknowledgement of trademarks

About Portrait Software

Portrait Software is now part of Pitney Bowes Software Inc.

Portrait Software enables organizations to engage with each of their customers as individuals, resulting in improved customer profitability, increased retention, reduced risk, and outstanding customer experiences. This is achieved through a suite of innovative, insight-driven applications which empower organizations to create enduring one-to-one relationships with their customers.

Portrait Software was acquired in July 2010 by Pitney Bowes to build on the broad range of capabilities at Pitney Bowes Software for helping organizations acquire, serve and grow their customer relationships more effectively. The Portrait Customer Interaction Suite combines world leading customer analytics, powerful inbound and outbound campaign management, and best-in-class business process integration to deliver real-time customer interactions that communicate precisely the right message through the right channel, at the right time.

Our 300 + customers include industry-leading organizations in customer-intensive sectors. They include 3, AAA, Bank of Tokyo Mitsubishi, Dell, Fiserv Bank Solutions, Lloyds Banking Group, Merrill Lynch, Nationwide Building Society, RACQ, RAC WA, Telenor, Tesco Bank, T-Mobile, Tryg and US Bank.

Pitney Bowes Software Inc. is a division of Pitney Bowes Inc. (NYSE: PBI).

For more information please visit: http://www.pitneybowes.co.uk/software/

UK	**America**	**Norway**
Portrait Software	Portrait Software	Portrait Software
The Smith Centre	125 Summer Street	Portrait Million Handshakes AS
The Fairmile	16th Floor	Maridalsveien. 87
Henley-on-Thames	Boston, MA 02210	0461 Oslo
Oxfordshire, RG9 6AB, UK	USA	Norway
Email: support@portraitsoftware.com	Email: support@portraitsoftware.com	Email: support@portraitsoftware.com
Tel: +44 (0)1491 416778	Tel: +1 617 457 5200	Tel: +47 22 38 91 00
Fax: +44 (0)1491 416601	Fax: +1 617 457 5299	Fax: +47 23 40 94 99

About this document

Purpose of document

To inform the intended audience on how to install the Portrait Foundation Software Development Kit (SDK) and how to set up and use the build environment.

The Portrait Foundation SDK replaces both the Portrait Build Kit and the Portrait Development Kit, and allows you to have a single installation that provides everything required to create project-specific implementation install sets.

Intended audience

Developers and Build Controllers creating project-specific implementations.

Related documents

ASP.NET Application Development Guide (NET_ADK_User_Guide.pdf)

Extending Applications using .NET (Portrait_NET_SDK.pdf)

Portrait Foundation Installation Guide

Software release

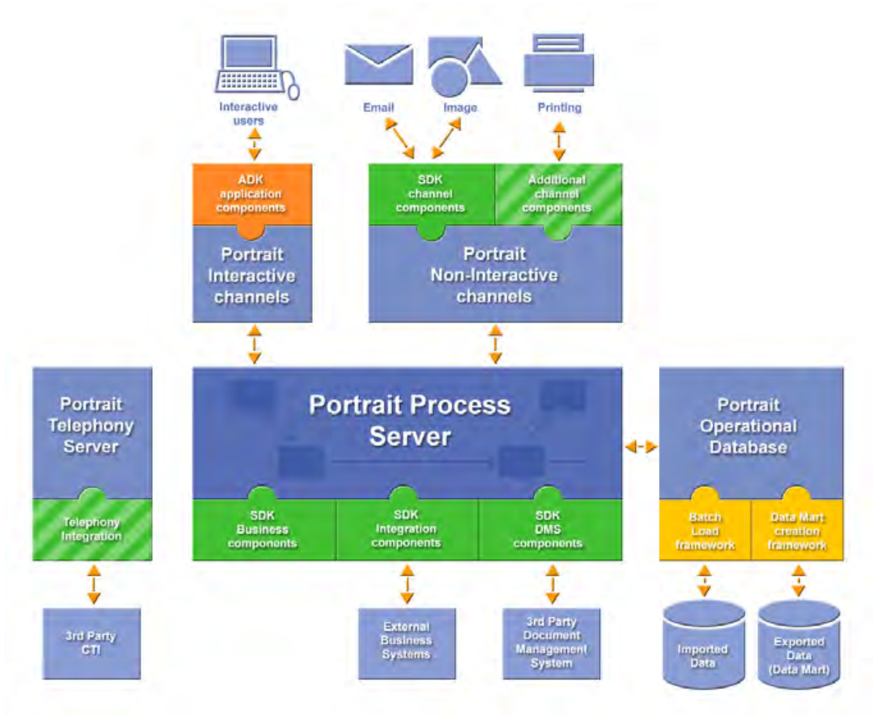
Portrait Foundation 5.0 or later.

Contents

1 Introduction

Portrait Foundation is a highly flexible, process-driven Customer Interaction Management system that has been designed to adapt to most business scenarios without the need to undertake development work. However, there are some situations in which it is necessary to expand upon the core functionality of the system, and Portrait Foundation offers several ways to extend this functionality.

Figure 1 – Portrait architecture


Interactive users
Email
Image
Printing
ADK application components
SDK channel components
Additional channel components
Portrait Interactive channels
Portrait Non-Interactive channels
Portrait Telephony Server
Telephony Integration
Portrait Process Server
SDK Business components
SDK Integration components
SDK DMS components
Portrait Operational Database
Batch Load framework
Data Mart creation framework
3rd Party CTI
External Business Systems
3rd Party Document Management System
Imported Data
Exported Data (Data Mart)

The Software Development Kit (SDK)

- Enables Developers to build new "plug-in" components that can be used in various parts of the system, including within Portrait Foundation process models
- Enables Developers to create and extend the user interfaces of Portrait Foundation applications such as the Contact Centre.

Additional Development Areas It is possible to extend Portrait Foundation in certain other areas that are not currently covered by the development kits. These areas include integration with third party telephony and printing solutions.

Importing and Exporting Data The Portrait Foundation operational database provides a flexible, metadata-driven schema to support Portrait's configurable business entities. Facilities are provided to import bulk data into the Portrait schema and to export data into an analytical database. Each of these facilities can be extended to meet the specific needs of a particular implementation.

2 Overview

2.1 Foundation Application Development

The Software Development Kit provides a rich, graphical environment for developers to create and extend Portrait Foundation applications such as the Contact Centre and Internet applications. Support for building applications using web-based technologies (ASP.NET) is provided through a set of Visual Studio templates.

The features and functionality supported by the SDK for Application development are summarised below.

- **Portrait Application Framework** – The framework consists of a set of interface definitions and base class implementations that provide a .NET accessible layer on top of the technology-independent Portrait Foundation components. This framework manages the common functionality at the Portrait Foundation presentation layer and offers a high level interface for producing new or customised Portrait Foundation applications based on .NET technology.
- **Foundation Web Application** – This template provides a starting point for a Portrait Foundation ASP.NET application based.
- **Foundation Web Package** – Similar to the Application template, the Package template provides a starting point for a Portrait Foundation package, which can be incorporated in a Portrait Foundation application.
- **Foundation Control Library** – A set of user interface controls derived from the standard .NET controls and accessible through the Visual Studio .NET Designer Toolbox. By using these controls, a Developer ensures a consistent look and feel and also benefits from simplified binding to data sourced from within Portrait Foundation.
- **Foundation Web Pages** – A set of templates to generate the starting point for different types of user interface that may be required within a Portrait Foundation ASP.NET application. This includes Custom UI pages, View pages and Internet pages as .aspx and .aspx.cs files containing classes derived from a set of Portrait Foundation base classes.
- **Foundation Web Control** – This template creates a new Portrait .NET Custom Controls, based on an existing .NET or Portrait Foundation control. The generated classes contain skeleton code for binding the control to Portrait Foundation data.
- **Localisation Add-in** – A tool for extracting and managing string resources to aid the process of translation and localisation of a Portrait Foundation application.

See the .NET Application Development Guide *(NET ADK User Guide)* for more details.

2.2 Extending Foundation Applications

The Portrait Software Development Kit provides a Visual Studio templates to enable developers to write new plug-in components using .NET (C#, Visual Basic .NET or any other .NET language).

The components supported by the SDK are summarised below:

- **Host Integration Framework Components** – The Host Integration Framework is a flexible mechanism for integrating Portrait Foundation with external systems. It provides an environment where various components (Transformers, Adapters, Data Access Transactions and Mapping Editors) can be plugged in as required. The purpose of these components is described in detail in the Host Integration Framework User Guide.
- **Custom Nodes** – Custom Nodes enable business-specific functionality to be encapsulated into a re-usable component. Once written and registered, a Custom Node may be dropped into any process model requiring that functionality. Custom Nodes are sometimes implemented as an alternative to Script Nodes or sub-models as they tend to be more efficient.
- **Simplex Channel Components** – Additional technologies may be incorporated into the inbound and outbound Simplex Channel infrastructure by developing new Simplex Channel components. These include inbound Channel Drivers (for retrieving an inbound document from a given source, such as a file server), outbound Email Drivers (for sending emails from a server other than MS Exchange) and Document Transports (as a replacement for the default MSMQ transports used within the Simplex Channels.).
- **Document Management System Accessors** – Accessors enable document retrieval, association and export functionality to be exposed from external document management systems. By providing a standard interface that is known to Portrait Foundation nodes, DMS Accessor components enable new document management systems to be accessed from within process models.
- **Encryption Provider –** Providers implement encryption algorithms for use by a Data Comparison Strategy. These are currently used in the Authentication Interaction Node.
- **Client Event Components** – The Client Events mechanism enables messages to be set to Portrait Foundation clients applications. The SDK provides support for developing new Subscriber and Resolver components, which are used to extend the server-side functionality of the events mechanism.

You can find information about how each component type fits within the Portrait Foundation architecture, and how to write, configure and test a new component in Extending Application using .NET *(Portrait NET SDK)*

2.3 Additional Development Areas

Portrait Foundation provides extensibility in two other areas – Computer Telephony Integration (CTI) and Simplex Channels. It is likely that development support will be provided in the SDK in a future release, but currently it is necessary for implementation teams to contact Portrait Support for assistance in extending either of these areas.

Telephony integration is summarised below. Additional information can be found in the Integrating Telephony Developers Guide.

Telephony – Integration with third party telephony systems used to be provided through the Portrait Telephony Server (PTS) and Portrait Telephony Client (PTC). A new architectural framework has been put in place that provides support for client-side telephony integration with Portrait Foundation ASP.NET applications (e.g. Contact Center). This approach is consistent with what telephony vendors are doing and is intended to be a replacement for the Portrait Telephony Server (PTS) integration.

The new implementation provides an out of the box client adapter for Genesys v8. This is the successor to the Genesys 7.2 TLib integration that was utilized by the Portrait Telephony Server (PTS) integration. The Genesys 8 integration is a client-side or desktop integration as opposed to a server-side integration which the PTS provided.

Contact your Portrait Support helpdesk should you have a specific requirement for integrating third party telephony systems.

2.4 Importing and Exporting Data

The Portrait Foundation operational database provides a flexible, metadata-driven schema to support Portrait's configurable business entities. Facilities are provided to import bulk data into the Portrait Foundation schema and to export data into an analytical database. Each of these facilities can be extended to meet the specific needs of a particular implementation.

Batch Loading Framework – Some Portrait Foundation implementations have a requirement to import bulk data into the Portrait Foundation database. For example, it may be necessary to regularly import engagements that take place through non-Portrait Foundation channels. Third party ETL (Extract Transform Load) tools are generally used to schedule and execute batch loads.

An ETL tool is responsible for acquiring data, transforming it and inserting it into the database. However, the metadata-driven schema in Portrait's operational database presents some challenges for the ETL tool. To overcome this, Portrait Foundation provides a Batch Load Framework to support the transformation of configurable business entities and reference data during the batch load process.

Contact your Portrait Support helpdesk should you have a specific requirement for batch loading.

Data Mart Creation – Portrait Foundation provides a mechanism for extracting data from the operational database into an offline data store known as the Data Mart. This data mart can be used directly for querying this data or it can be used as a staging area between Portrait Foundation and an organisation's existing data warehouse or analysis database.

A Data Mart creation utility, supplied with the base Portrait Foundation product, provides a means to build and populate a database whose structure is defined by the business configuration data. The creation utility has an Extension Framework that enables implementation-specific processing to be incorporated into the Data Mart build and population processes.

The Data Mart Creation Utility User Guide provides details of how to use the Extension Framework to incorporate implementation-specific processing.

3 Foundation SDK installation

The Portrait Foundation SDK provides the following features:

Visual Studio Templates: Provides a set of Visual Studio project and item templates and C# components enabling developers to build new C# "plug-in" components and extend Portrait Foundation ASP.NET applications.

Build Environment: The build environment will give you access to the Build Environment Wizard which allows you to create build environments and the Build Control Tool which allows you to add files and projects to your project specific implementation build.

MSI Implementation Install: Provides the building blocks required by customers to create MSI based Implementation install sets without the need of an InstallShield licence.

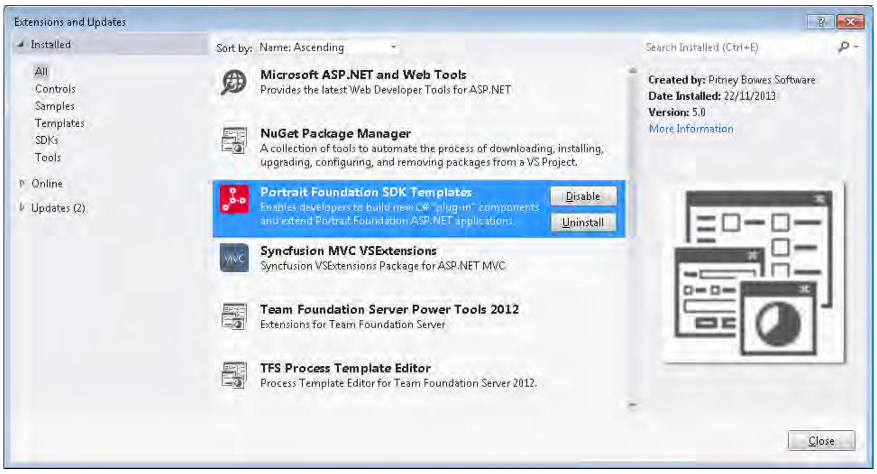
C++ SDK: The C++ SDK contains the library and header files required to build any legacy C++ code nodes and plug-ins.

The install is now part of the Core Software install which can be found on the release media under **Software\Installsets\Core_Software.**

For more details please refer to the Portrait Foundation Installation Guide.

Visual Studio templates

The Portrait Foundation SDK Templates are installed as a Visual Studio extension which allows developers to easily disable or uninstall them.


Extensions and Updates

Installed: All, Controls, Samples, Templates, SDKs, Tools; Online; Updates (2)

Sort by: Name: Ascending

Search Installed (Ctrl+E)

Microsoft ASP.NET and Web Tools
Provides the latest Web Developer Tools for ASP.NET

NuGet Package Manager
A collection of tools to automate the process of downloading, installing, upgrading, configuring, and removing packages from a VS Project.

Portrait Foundation SDK Templates
Enables developers to build new C# "plug-in" components and extend Portrait Foundation ASP.NET applications.
Disable | Uninstall

Syncfusion MVC VSExtensions
Syncfusion VSExtensions Package for ASP.NET MVC

Team Foundation Server Power Tools 2012
Extensions for Team Foundation Server

TFS Process Template Editor
Process Template Editor for Team Foundation Server 2012.

Created by: Pitney Bowes Software
Date Installed: 22/11/2013
Version: 5.0
More Information

Close

NB: After the installation of the Portrait Foundation SDK templates, users may need to enable the use of them in Visual Studio. After the Core Software installation has completed do the following:

- Close any open instances of Visual Studio.
- Open a new instance of Visual Studio and select "Extensions and Updates…" from the **Tools** menu.
- Select "Portrait Foundation SDK Templates" and click "Enable".

4 Build Environment

4.1 Build Environment Setup

The build environment is intended to be used by projects to generate Portrait Foundation Implementation installs. The processes described below are designed to run on a designated build machine.

4.1.1 Setting up the build machine

The software on the build machine must be set up in accordance with the guidelines detailed in section 3 Foundation SDK installation.

The typical structure for a Portrait Foundation implementation project may be:

- MyFolder
    - MySourceCode
        - MyProject
            - Code
                - CRMExtensions
                - WebApplication
            - Database
            - Repository

It is recommended that the L drive is used so that file links within the installation program do not have to be updated. To associate a drive to the source code folder, use the **subst** command. For example:

```
subst L: D:\MyFolder\MySourceCode
```

In this example the Build Environment Location would be **L:\MyProject**

Run the Build Environment Wizard to configure *setvariables.bat* and copy the relevant files into your build folder structure. On completion the folder structure under the L drive should look something like this:

- MyProject
    - **Build**
    - Code
        - CRMExtensions
        - WebApplication
    - **Common**
    - Database
    - **MSI**
    - Repository

Finally use the Build Configuration Tool to edit the other customizable files.

NB: It is recommended not to add these folders (Build, Common and MSI) into source control as they are likely to change in future releases. If you wish to store any customizable files in source control, create a separate folder (e.g. Build_Files) and create a batch files to copy these files into the correct folders once the Build Environment Wizard has been run.

4.1.2 Build Environment Wizard

The Build Environment Wizard is used to create build environments that will help when building project-specific implementation install sets. Through this wizard you will be able to create, edit and delete build environments.

Creating a build environment

1. Run the Build Environment Wizard.

   The wizard can be run either during the Portrait Foundation SDK Setup or by double-clicking the **Build Environment Wizard** shortcut in the **Portrait Foundation > SDK** group on the Start menu.

2. On the "Manage Build Environments" screen, select **Create a new Build Environment** and click **Next**.

3. On the "Choose a Location for the Build Environment" screen, enter a new location — or click **Browse** and select a new location — for the build environment and click **Next**.

4. On the "Choose a location install target" screen, enter a new location — or click **Browse** and select a new location — for the build environment and click **Next**.

5. On the "Select the Repository and Workspace locations" screen, set the repository settings for the build process. This is the repository to which components and web applications will be released, and from which the install sets will be created.

   - The **Repository location** must contain a `*.policy` file to validate that you are pointing to an actual repository.

   - The **Workspace location** has to contain a `*.pws` file to be able to add custom files to the repository's workspace.

6. Click **Next** to display the final screen, and click **Finish** to create the build environment.

Editing a build environment

Following installation, build environment settings can be modified using the Build Environment Wizard:

1. From the Start menu, choose the **Portrait Foundation > SDK** group and select **Build Environment Wizard**.

2. In the "Manage Build Environments" screen, select **Edit an Existing Build Environment** and click **Next**.

3. On the "Select an existing Build Environment" screen, choose the environment you want to edit and click **Next**.

Subsequent screens in the wizard are displayed in the same order as when creating a build environment (see *Creating a build environment.*)

Deleting a build environment

If required, a previously-created build environment can be removed from your PC:

1. From the Start menu, choose the **Portrait Foundation > SDK** group and select **Build Environment Wizard**.

2. In the "Manage Build Environments" screen, select **Delete an Existing Build Environment** and click **Next**.

3. On the "Select an existing Build Environment" screen, choose the environment you want to delete and click **Next**.

4. To confirm deletion of the selected build environment, click **Finish**.

4.1.3 Build Configuration Tool

The Build Configuration Tool (BCT) is used to add, edit or remove project specific files, folders and assemblies that will be built as part of your build process. To be able to run this wizard you must already have a pre-existing build environment that has been created using the Build Environment Wizard.

1. Run the Build Configuration Tool.

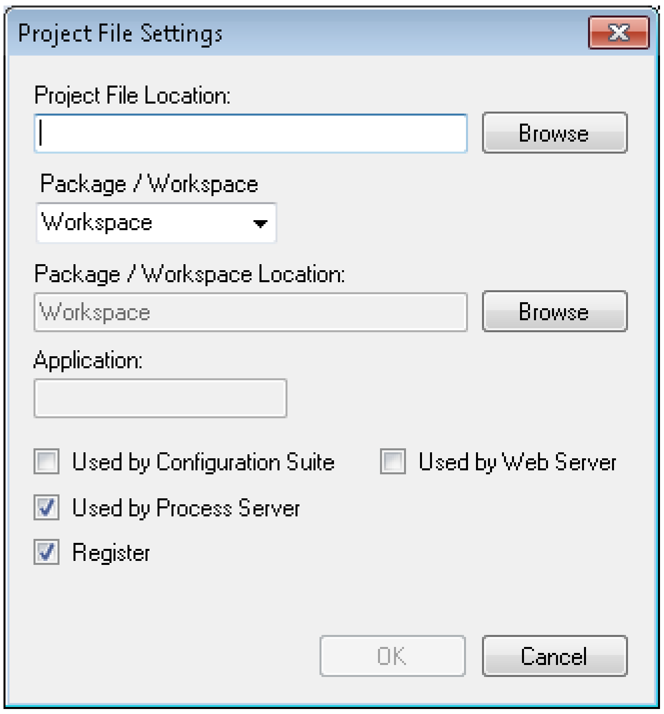
   You can launch the Build Configuration Tool from the Build Environment Wizard when you have created a new build environment. Alternatively, select the **Build Configuration Tool** shortcut in **Portrait Foundation > SDK** group on the Start menu.

2. On the "Select an existing Build Environment" screen, select a build environment — or click **Browse** to select a build environment location and click **Next**.

   Note that if you browse to select a build environment, you must select the **Build** folder for the build environment.

3. Use "Select projects to add to the build" screen to add, edit or delete projects in the build environment:

   a To add a project click **Add**. The Project File Settings dialog displays:



Project File Settings

Project File Location:

Package / Workspace: Workspace

Package / Workspace Location: Workspace

Application:

☐ Used by Configuration Suite ☐ Used by Web Server

☑ Used by Process Server

☑ Register

   b Click **Browse** to locate a project file, which must be of the form `*.csproj` or `*.vcxproj`

   c Select the type of build you require, either **Project** or **Workspace**.

Most projects will be built as part of the repository owner's workspace. If the project is part of a specific package: select **Package**, browse to the location of the package and choose the `*.ppk` project definition file.

d Select the project options:

If the project is to be available for use in the Portrait Configuration Suite, select **Used by Configuration Suite**.

If the project represents a DAT or Code Node that needs to be registered on the process server then select **Used by Process Server** and **Register**.

If the project represents an Application Assembly that needs to run on the web server then select **Used by Web Server**. When you select this option you also need to supply the name of the Application; possible values include:

- ContactCentre.Net
- ContactCentreAuth.Net
- SCProductManager
- SCCampaignsManager

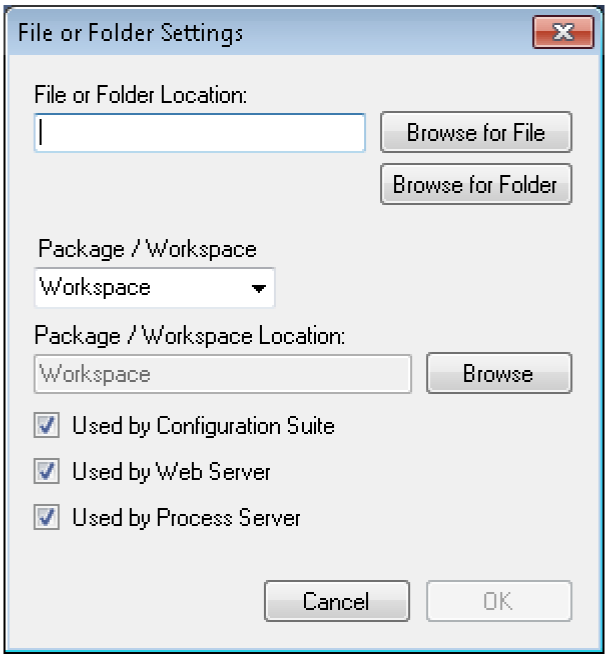
e To edit a project that is included in the build process, select a project in the grid and click **Edit**. In the Project File Settings dialog make any required changes and click **OK**.

f To remove a project from the build process, select a project in the grid and click **Delete**. Note that only the record of the project will be deleted, any physical files will remain on the build machine and must be manually deleted.

4. When all the required options have been chosen, click **OK** to close the Project File Settings dialog and click **Next**.

5. Use the "Select files and folders to add to the build" screen to add, edit or delete files and folder that will be included in your implementation's install set:

a To add a file or folder to the project click **Add**. The File or Folder Settings dialog displays:


File or Folder Settings

File or Folder Location:

Package / Workspace: Workspace

Package / Workspace Location: Workspace

☑ Used by Configuration Suite

☑ Used by Web Server

☑ Used by Process Server

b Click **Browse for File** to add one or more files to the project

c Click **Browse for Folder** to add a folder (and all its subfolders) to the project.

d Select the type of build you require, either **Project** or **Workspace**.

Most projects will be built as part of the repository owner's workspace. If the project is part of a specific package: select **Package**, browse to the location of the package and choose the `*.ppk` project definition file.

e Select the file or folder options:

If the project is to be available for use in the Portrait Configuration Suite, select **Used by Configuration Suite**.

If the project is to be used with the web server or process server, select **Used by Web Server** or **Used by Process Server** respectively.

f To edit a file or folder that is already include as part of the build process, select the entry in the grid and click **Edit**. In the File or Folder Settings dialog make any required changes and click **OK**.

g To remove a file or folder from the build process, select the entry in the grid and click **Delete**. Note that only the record of the file or folder will be deleted, any physical files or folders will remain on the build machine and must be manually deleted.

6. When all the required options have been chosen, click OK to close the File or Folder Settings dialog and click **Next**.

7. On the final screen, click **Finish** to commit your changes to the build environment.

4.2 Implementation Build Process

To build an implementation you need to run the **ProcessMain.bat** file, which in turn runs a collection of additional batch files. **ProcessMain.bat** can take different parameters, covering the different variations of build that may be required.

Before you run **ProcessMain.bat**, make sure that there are no missing references in your Visual Studio project files, so that the component builds correctly. Customers using a source code provider may also want to get the latest code before kicking off the build process.

Command line parameters

An MSI Implementation install will be generated if this batch file is run without passing in any command line parameters.

```
ProcessMain.bat
```

Running this command performs the following actions:

- Compile the components and applications
- Copy all components and web files into the repository
- Extract all the files referenced by the workspace
- Generate the file structure required by the MSI install

To update the relevant version files and increment the build number in ait_version.cfg, use the **incrementversion** parameter. By default the build number is not incremented.

```
ProcessMain.bat incrementversion
```

To rebuild the components with generating any MSI output, use the **buildonly** parameter.

```
ProcessMain.bat buildonly
```

To get the latest source code from version control, use the **getlatest** parameter.

```
ProcessMain.bat getlatest
```

4.2.1 Project implementation versioning

Each Portrait Foundation release has a version number, such as 1.1.0. The versioning of an implementation project uses this and extends it by adding an implementation build number to the end. Following this pattern, an implementation version could be 1.1.14.12.

This implementation version number is incremented during the build process and stored in the **ait_version.cfg**, **ait_version.h** and **AITVersionInfo.cs** files.

The **ait_version.cfg** file stores the version number and is used within the build process to set-up the versions in the other files. This file is also included in the non MSI Install to provide the version of the installed application. The build number is only incremented automatically when running ProcessMain.bat with no parameters. When creating an MSI install, this must be updated manually.

The **ait_version.h** file is updated by the build process and provides the version that is displayed in the properties of the built components. This version is built into a component using a version block in the component's RC file. The **AITVersionInfo.cs** file contains version information for all C# assemblies and if required should be referenced by all .csproj projects.

The implementation install program can be run on a target machine (development machine, test environment or production server) once the base Portrait Foundation install program has been completed.

4.2.2 Customizable files

The build/install process can be customized by changing the following configuration text files:

- **ait_version.cfg** – contains the version numbers used by the build/install process. This file can be edited using Windows Notepad.
- **components.txt** – contains all the Visual Studio project files to be built by the build process. This should be edited using the Build Configuration Tool.
- **exclude.txt** - contains a list of files to be excluded during the web application copy steps. If you wish to exclude additional file extensions, this file can be edited using Windows Notepad.
- **files.txt** – contains a list of files to be copied into the Repository and Install set by the build process. This can be edited using the Build Configuration Tool.
- **requiredwebapps.txt** – contains a list of additional applications that should be included when the install set is generated. This file can be edited using Windows Notepad.

The Major, Minor and ServicePack version numbers can be changed by editing **ait_version.cfg**. The ProjectBuildNo is automatically incremented by the build process, if it does not exist then the initial value will be 1.

The build process uses the **components.txt** file to define which applications to include in the install set. So if you have a CSPROJ that is "Used by Web Server" and the Application is **ContactCentre.Net**, this application will automatically be added to the install set.

If you wish include an Application that you are not building because it is provided in a package by a third party. Then edit **requiredwebapps.txt** and add the Application name, this should match the folder name under "_install\Release\WebServer\WebApps" of the third party package in your Repository (e.g. Edge2020). Each name should be added to a separate line in the test file. If you wish to use a third party application, the relevant package must be included in your workspace.

4.2.3 The build process batch files

The build batch files are stored in the build directory, under the Build Environment Location that is setup by the Build Environment Wizard.

ProcessMain.bat	Controls the implementation build process. This calls each of the other batch files in turn to set the version number, build the components and web application, copy the package contents, assemble all package contents and build the debug and production install sets.
BuildComp.bat	Builds each component in the component list.
CopyComp.bat	Copies the built components into the repository.
CopyFile.bat	Copies a file or group of files into the repository.
CopyRep.bat	Copies the relevant parts of the repository into the staging area prior to building the install set.
ReportAndLog.bat	Echoes log messages to standard output and log files.
SetVariables.bat	Contains all the information required by the other batch files for the build process to operate. This should only be edited by the Build Environment Wizard.
SetVersion.bat	Sets and increments the project build version using inimod.exe in the version files: ait_version.cfg, ait_version.h & AitVersionInfo.cs.
update_source_code.bat	Gets the latest files from version control.

4.2.4 Strong-Name signing

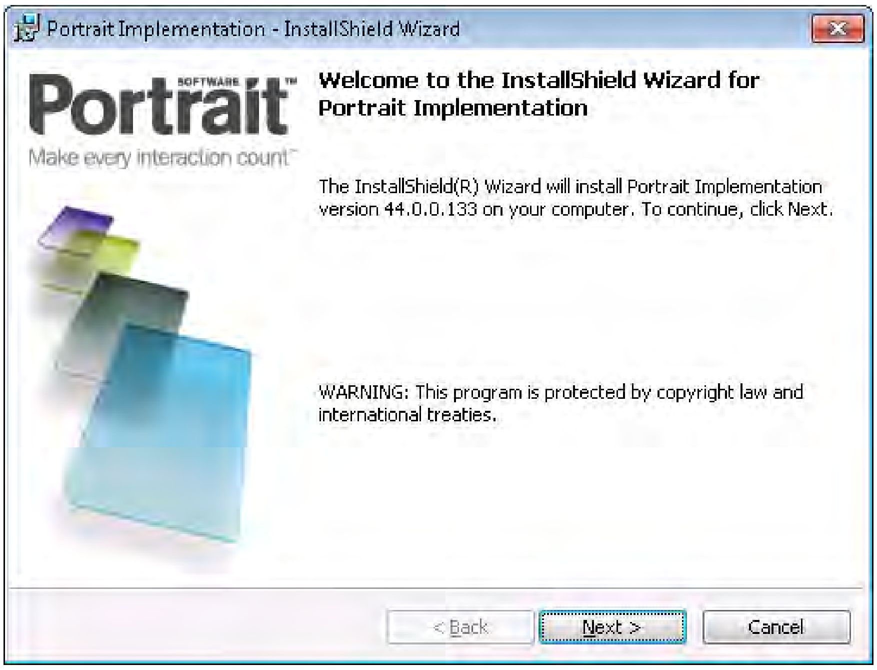
The Foundation SDK Build Environment no longer provides any key files (.snk) and the references in AITVersionInfo.cs have been removed. It is recommended that all customers generate their own key files and ensure that all their Visual Studio projects are signed using this key file.

4.3 MSI Based Implementation Install

4.3.1 Implementation MSI

The Portrait Foundation Build Environment provides a basic "Portrait Implementation.msi" that contains the following out of the box features. The default install location for this new install is "**<ProgramFiles>[1]\PST\Portrait Implementation**".

1. Welcome screen with customizable product name and version number.



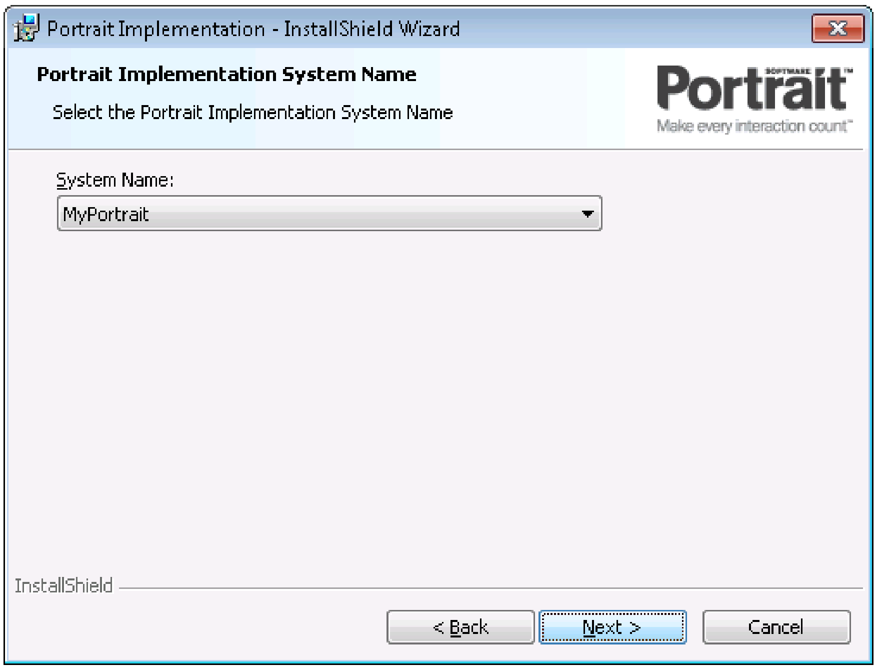
Portrait Implementation - InstallShield Wizard

Welcome to the InstallShield Wizard for Portrait Implementation

The InstallShield(R) Wizard will install Portrait Implementation version 44.0.0.133 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

2. Multi-tenant system selection.



Portrait Implementation - InstallShield Wizard

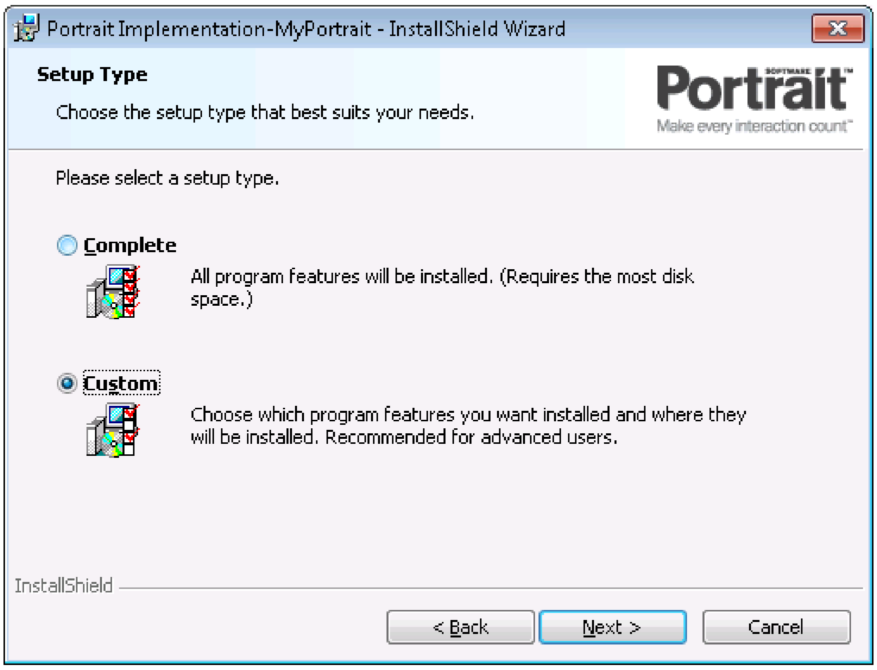
Portrait Implementation System Name

Select the Portrait Implementation System Name

System Name: MyPortrait

[1] The path for <ProgramFiles> may differ depending on the version of operating system and how it has been setup. Typically on a 32-bit version of Windows this would be "C:\Program Files" and on a 64-bit version "C:\Program Files (x86)".

3. Complete and customer setup types.


Portrait Implementation-MyPortrait - InstallShield Wizard
Portrait SOFTWARE™
Make every interaction count™
InstallShield
< Back
Next >
Cancel

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

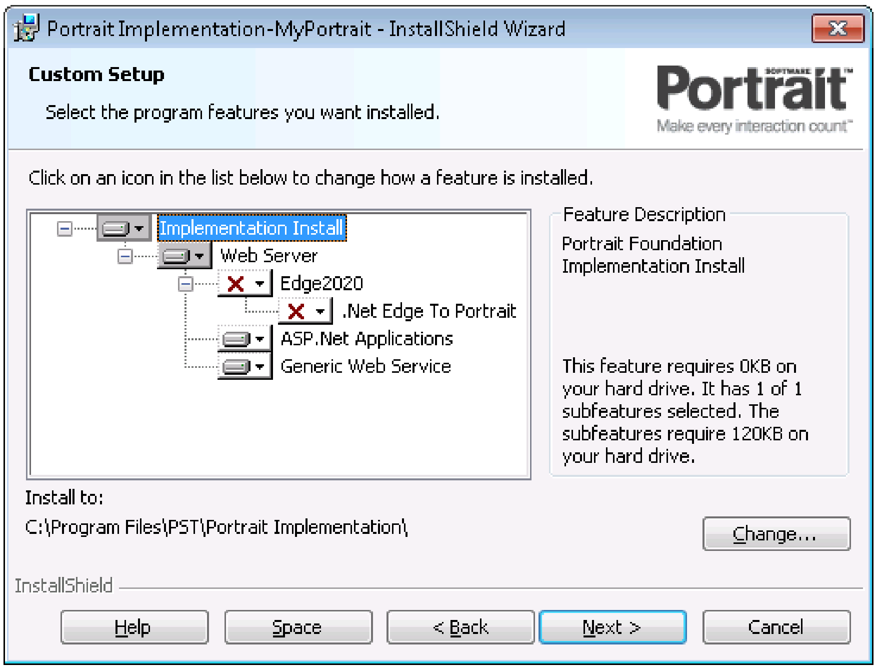
( ) Complete

All program features will be installed. (Requires the most disk space.)

[✓ Custom]

Choose which program features you want installed and where they will be installed. Recommended for advanced users.

4. Standard Web Server install for the following features.


Portrait Implementation-MyPortrait - InstallShield Wizard
Portrait SOFTWARE™
Make every interaction count™
InstallShield
Change...
Help
Space
< Back
Next >
Cancel

Custom Setup

Select the program features you want installed.

Click on an icon in the list below to change how a feature is installed.

- Implementation Install
  - Web Server
    - ✗ Edge2020
      - ✗ .Net Edge To Portrait
    - ASP.Net Applications
    - Generic Web Service

Feature Description

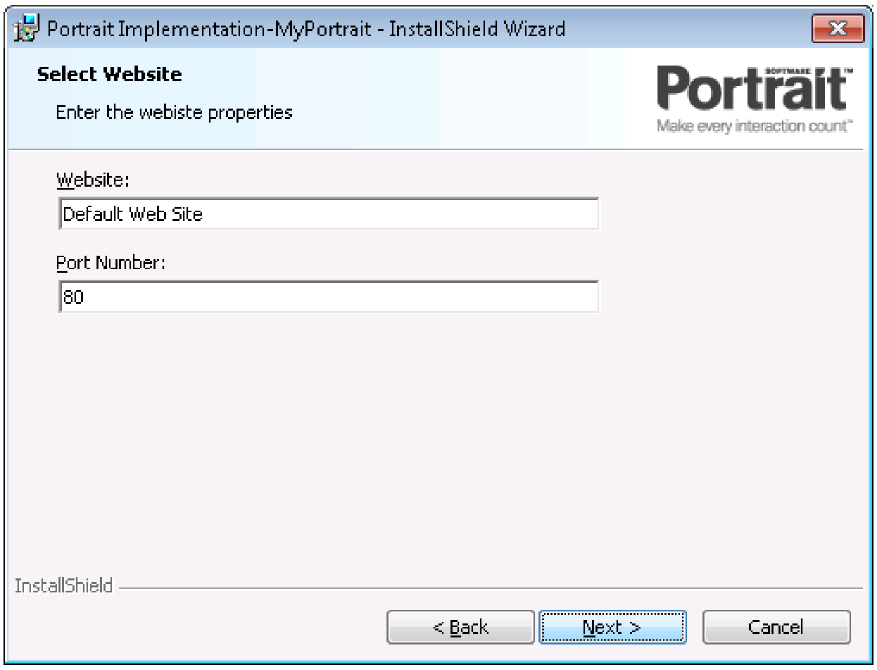
Portrait Foundation Implementation Install

This feature requires 0KB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 120KB on your hard drive.

Install to:

C:\Program Files\PST\Portrait Implementation\

5. Creation of the Application pool and Virtual directories required by these features.

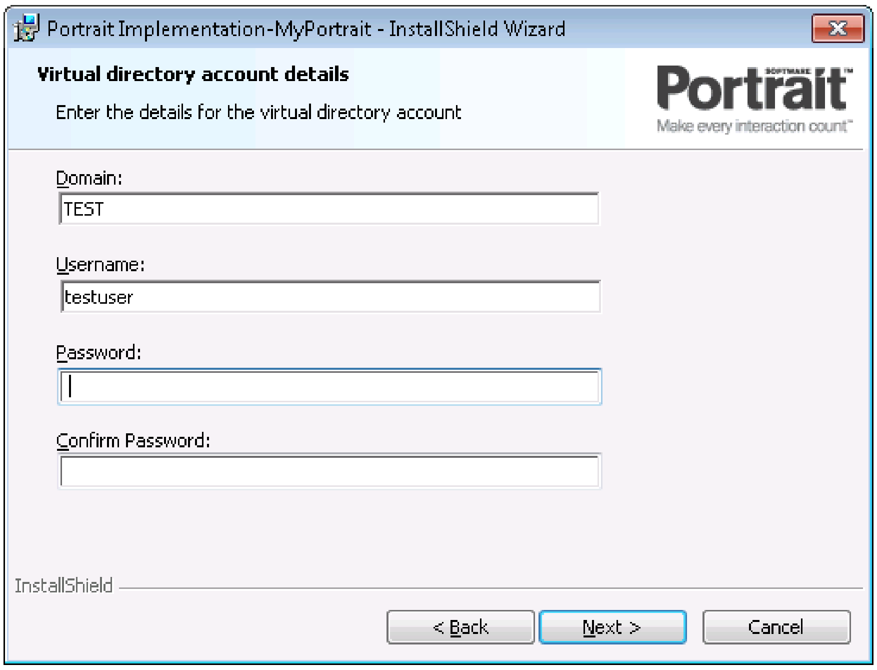

Portrait Implementation-MyPortrait - InstallShield Wizard

Select Website

Enter the webiste properties

Website: Default Web Site

Port Number: 80


Portrait Implementation-MyPortrait - InstallShield Wizard

Virtual directory account details

Enter the details for the virtual directory account

Domain: TEST

Username: testuser

Password:

Confirm Password:

4.3.2 MSI Transform

This MSI has been provided as a building block to allow customers to easily extend and create their own Implementation MSI install. This is achieved by creating a Transform (.mst) that is applied to the "Portrait Implementation.msi" to create an Implementation specific MSI.

Please note that a basic knowledge of how MSI installs are authored is required.

There are various tools available for creating and editing **.mst** files

- InstallShield (http://www.installshield.com/)
- InstEd (free from http://www.instedit.com/)
- Microsoft Orca[2]

This document provides examples of how to edit the MSI database tables directly using **InstEd**.

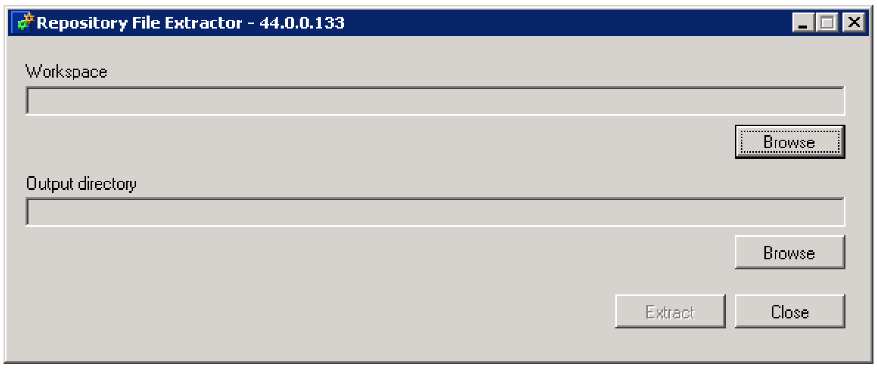
4.3.3 Creating MST files

The files required to create an MSI Transform can be found in the **MSI** folder of your Build Environment and has the following structure.

- MSI
  - CommonFiles
  - Examples
  - Transform

CommonFiles

This folder contains the tools used by the build process to generate the files required to create an MSI Transform. The Repository File Extractor tool (FdnRepositoryExtractor.exe) is run silently by one of the batch files, but can also be run as a Windows application. It will extract all the Web server and Process server components that are stored in the Packages that are included by a specific Workspace of a Portrait Foundation Configuration Repository.



Examples

This folder contains two example MSI Transform files.

- **All Applications.mst** – Used to create the All Application MSI install that accompanies this release. Providing examples of

[2] See Windows Installer development tools: http://msdn.microsoft.com/en-us/library/windows/desktop/aa372834(v=VS.85).aspx

- Ensuring that the Portrait Foundation Web Server component is installed.
- Adding a new dialog to capture Telephony settings.
- Installing various ASP.NET applications.

- **Interaction Optimizer.mst** – Used to create the IO 2.4 MSI install that is built against this release. Providing examples of
    - Installing Process server components.

Please use these for reference when creating you transform.

Transform

This folder should contain all the files required to create an MSI Transform once the build process has been run. See section 4.2 for more details.

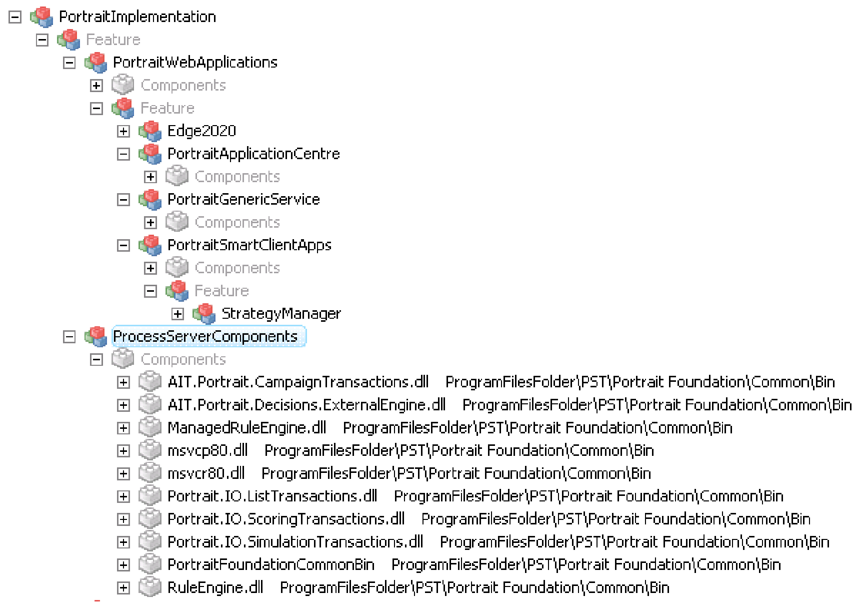
If the Build Environment has been configured correctly and the build process has completed successfully, there should be a "**Program Files**" folder that contains all the files that need to be installed by your Implementation MSI install.

To create your transform file using **InstEd** perform the following steps:

1. Launch **InstEd** and select File > Open.
2. Navigate to the **Transform** folder and select "Portrait Implementation.msi" and click Open.
3. Select Transform > New Transform
4. Enter a name and click Save.

4.3.3.1 Basic MSI concepts

The following relationships exist within a basic MSI.



```
⊟ PortraitImplementation
  ⊟ Feature
    ⊟ PortraitWebApplications
      ⊞ Components
      ⊟ Feature
        ⊞ Edge2020
        ⊟ PortraitApplicationCentre
          ⊞ Components
        ⊟ PortraitGenericService
          ⊞ Components
        ⊟ PortraitSmartClientApps
          ⊞ Components
          ⊟ Feature
            ⊞ StrategyManager
    ⊟ ProcessServerComponents
      ⊟ Components
        ⊞ AIT.Portrait.CampaignTransactions.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ AIT.Portrait.Decisions.ExternalEngine.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ ManagedRuleEngine.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ msvcp80.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ msvcr80.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ Portrait.IO.ListTransactions.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ Portrait.IO.ScoringTransactions.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ Portrait.IO.SimulationTransactions.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ PortraitFoundationCommonBin   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
        ⊞ RuleEngine.dll   ProgramFilesFolder\PST\Portrait Foundation\Common\Bin
```

- A **file** is linked to a **component** in the File table.
- A **component** is linked to a **directory** in the Directory table.
- A **feature** can have a parent **feature** which is defined in the Feature table.
- A **component** is linked to a **feature** in the FeatureComponent table.

Entries should therefore be added in the following order.

1. Directory
2. Component
3. Feature
4. FeatureComponent
5. File
6. Media

For further details of the MSI database tables please refer to:

http://msdn.microsoft.com/en-us/library/windows/desktop/aa368259(v=VS.85).aspx

General rules

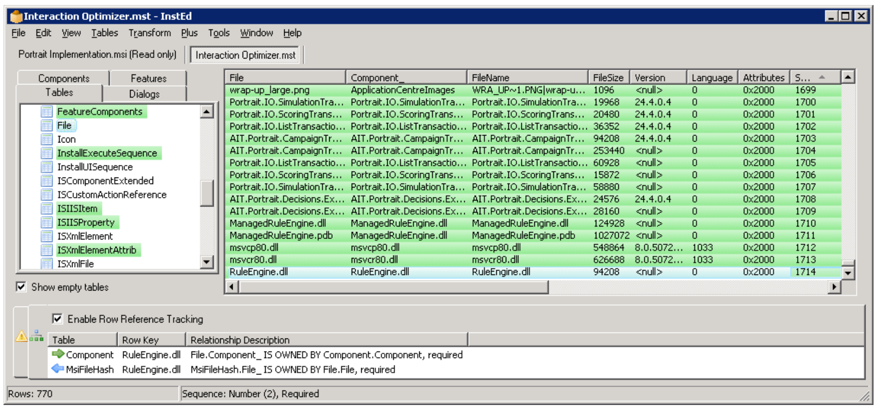
To change the name displayed by the install, edit the ProductName and ORIGINALPRODUCTNAME entries in the Property table.

Use a general component for web pages, xslt and xml files.

Use a specific component for special files like assemblies and executables.

When adding a file set the Attributes to "0x2000" (Noncompressed). Also keep a note of the sequence number and always use the next value.

Use the "Retrieve file details from external source path" context menu option to populate the other properties such as FileSize, Version and Language.

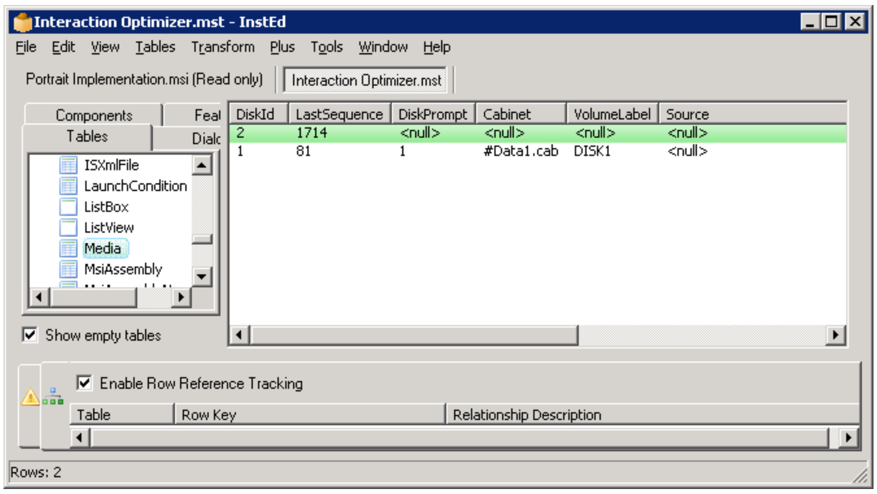


File	Component_	FileName	FileSize	Version	Language	Attributes	S...
wrap-up_large.png	ApplicationCentreImages	WRA_UP~1.PNG|wrap-u...	1096	<null>	0	0x2000	1699
Portrait.IO.SimulationTra...	Portrait.IO.SimulationTra...	Portrait.IO.SimulationTra...	19968	24.4.0.4	0	0x2000	1700
Portrait.IO.ScoringTrans...	Portrait.IO.ScoringTrans...	Portrait.IO.ScoringTrans...	20480	24.4.0.4	0	0x2000	1701
Portrait.IO.ListTransactio...	Portrait.IO.ListTransactio...	Portrait.IO.ListTransactio...	36352	24.4.0.4	0	0x2000	1702
AIT.Portrait.CampaignTr...	AIT.Portrait.CampaignTr...	AIT.Portrait.CampaignTr...	94208	24.4.0.4	0	0x2000	1703
AIT.Portrait.CampaignTr...	AIT.Portrait.CampaignTr...	AIT.Portrait.CampaignTr...	253440	<null>	0	0x2000	1704
Portrait.IO.ListTransactio...	Portrait.IO.ListTransactio...	Portrait.IO.ListTransactio...	60928	<null>	0	0x2000	1705
Portrait.IO.ScoringTrans...	Portrait.IO.ScoringTrans...	Portrait.IO.ScoringTrans...	15872	<null>	0	0x2000	1706
Portrait.IO.SimulationTra...	Portrait.IO.SimulationTra...	Portrait.IO.SimulationTra...	58880	<null>	0	0x2000	1707
AIT.Portrait.Decisions.Ex...	AIT.Portrait.Decisions.Ex...	AIT.Portrait.Decisions.Ex...	24576	24.4.0.4	0	0x2000	1708
AIT.Portrait.Decisions.Ex...	AIT.Portrait.Decisions.Ex...	AIT.Portrait.Decisions.Ex...	28160	<null>	0	0x2000	1709
ManagedRuleEngine.dll	ManagedRuleEngine.dll	ManagedRuleEngine.dll	124928	<null>	0	0x2000	1710
ManagedRuleEngine.pdb	ManagedRuleEngine.dll	ManagedRuleEngine.pdb	1027072	<null>	0	0x2000	1711
msvcp80.dll	msvcp80.dll	msvcp80.dll	548864	8.0.5072...	1033	0x2000	1712
msvcr80.dll	msvcr80.dll	msvcr80.dll	626688	8.0.5072...	1033	0x2000	1713
RuleEngine.dll	RuleEngine.dll	RuleEngine.dll	94208	<null>	0	0x2000	1714

Table	Row Key	Relationship Description
Component	RuleEngine.dll	File.Component_ IS OWNED BY Component.Component, required
MsiFileHash	RuleEngine.dll	MsiFileHash.File_ IS OWNED BY File.File, required

Rows: 770
Sequence: Number (2), Required

The uncompressed files structure makes it easier to manage automating the build process once the Transform has been created. The Media table only requires one additional entry, which covers all the files that have been added by the Transform.


Interaction Optimizer.mst - InstEd
File Edit View Tables Transform Plus Tools Window Help
Portrait Implementation.msi (Read only)
Interaction Optimizer.mst
Components
Feat
Tables
Dialc
ISXmlFile
LaunchCondition
ListBox
ListView
Media
MsiAssembly
Show empty tables
Enable Row Reference Tracking
Table
Row Key
Relationship Description

DiskId	LastSequence	DiskPrompt	Cabinet	VolumeLabel	Source
2	1714	<null>	<null>	<null>	<null>
1	81	1	#Data1.cab	DISK1	<null>

Rows: 2

4.3.3.2 Web server components

The files found under "**Transform\Program Files\PST\Portrait Implementation**" are the Web server components.

Within the "Portrait Implementation.msi" that are a set of predefined Directories, Components and Features.

Components

- ApplicationCentreWebFiles
    - \Application Centre\images
- ApplicationCentreXmlFiles
    - \Application Centre\xml
- ApplicationCentreXslFiles
    - \Application Centre\xsl
- ApplicationCentreHelpFiles
    - \Application Centre\help
- ApplicationCentreImages
    - \Application Centre\images
- ApplicationCentreIncludes

- \Application Centre\includes

- ApplicationCentreJQueryFiles
    - \Application Centre\includes\jQuery
- ApplicationCentreJQueryPluginsFiles
    - \Application Centre\includes\jQuery\Plugins
- ApplicationCentreJQueryCssFiles
    - \Application Centre\includes\jQuery\css
- ApplicationCentreJQueryBreadCrumbFiles
    - \Application Centre\includes\jQuery\css\BreadCrumb
- ApplicationCentreJQueryContextMenuFiles
    - \Application Centre\includes\jQuery\css\ContextMenu
- ApplicationCentreJQueryLayoutFiles
    - \Application Centre\includes\jQuery\css\Layout
- ApplicationCentreJQueryMenuRibbonFiles
    - \Application Centre\includes\jQuery\css\MenuRibbon
- ApplicationCentreJQueryTreeViewFiles
    - \Application Centre\includes\jQuery\css\TreeView
- ApplicationCentreJQueryCupertinoFiles
    - \Application Centre\includes\jQuery\css\cupertino
- ApplicationCentreJQueryCupertinoImages
    - \Application Centre\includes\jQuery\css\cupertino\images

Directories

- DIR_SYSTEM_DEFAULT_WEB_APPCENTRE_BIN
    - \Application Centre\bin

Features

- PortraitWebApplications
    - PortraitApplicationCentre

Adding Assemblies

Add a new row to the Component table. Use the name of the assembly for the new component and select the DIR_SYSTEM_DEFAULT_WEB_APPCENTRE_BIN directory. Set the Attributes to "0x0008" and the KeyPath to the name of the assembly.

Add a new row to the File table for the assembly and select the component created previously. If there is a PDB file this can also be added to the same component.

Map an entry in the FeatureComponents table, set the feature to "PortraitApplicationCentre" and select the new component.

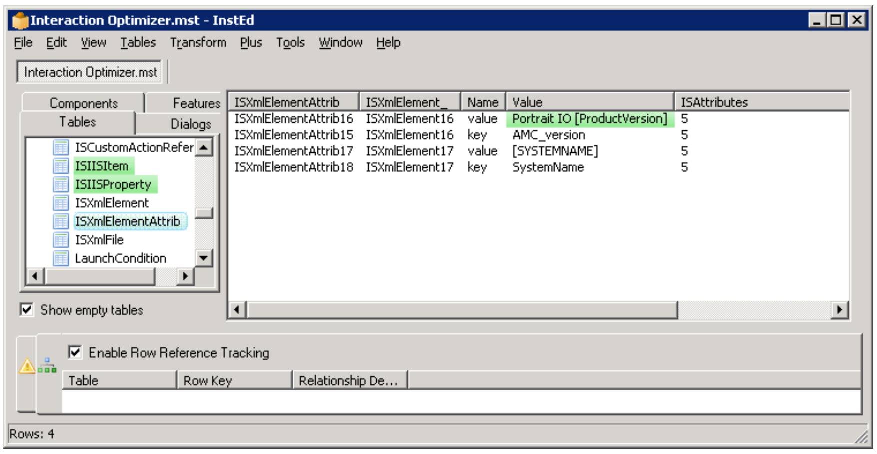
Adding other Application Centre web files

Add a new row to the File table and select the component that is related to the installation folder.

Web.config

If your project contains implementation specific changes to the web.config file, then find the "web.config" file assigned to the "ApplicationCentreWeb.config" component in the File table and change the Attributes to "0x2000". This will make the install use your uncompressed version of this file.

To change the product name and version in web.config, select the ISXmlElementAttrib table and edit the Value for ISXmlElementAttrib16.



ISXmlElementAttrib	ISXmlElement_	Name	Value	ISAttributes
ISXmlElementAttrib16	ISXmlElement16	value	Portrait IO [ProductVersion]	5
ISXmlElementAttrib15	ISXmlElement16	key	AMC_version	5
ISXmlElementAttrib17	ISXmlElement17	value	[SYSTEMNAME]	5
ISXmlElementAttrib18	ISXmlElement17	key	SystemName	5

4.3.3.3 Process server components

The files found under "**Transform\Program Files\PST\Portrait Foundation**" are the Process server components.

To create the relevant Process server features, components and directories do the following:

- Add a new Feature called "ProcessServerComponents" and set the Feature_Parent to "PortraitImplementation".
- Create the relevant Portrait Foundation Directories, see "Interaction Optimizer.mst".
- Add a new Component called "PortraitFoundationCommonBin" and set the directory to "DIR_PST_PORTRAIT_FOUNDATION_COMMON_BIN".
- Add a new ControlEvent for the "SetupType" dialog to remove the "ProcessServerComponents" feature from the install if FOUNDCRMSERVER<>"TRUE", see "Interaction Optimizer.mst".
- Add an entry to the CreateFolder table, set the directory to "DIR_PST_PORTRAIT_FOUNDATION_COMMON_BIN" and the components to "PortraitFoundationCommonBin".
- Add a new CustomAction called "setFdnInstallDirectory", see "Interaction Optimizer.mst".
- Edit the InstallExecuteSequence table and add entries for "setFdnInstallDirectory", see "Interaction Optimizer.mst".
- Add a new Property called "INSTALLFOLDER", see "Interaction Optimizer.mst".
- Add two new entries to the AppSearch table for the "INSTALLFOLDER" property, see "Interaction Optimizer.mst".

- Add new entries to the RegLocator table for "FdnInstallFolder32" and "FdnInstallFolder64", see "Interaction Optimizer.mst".

Finally add the relevant Process server components to the install. All these files will be located under "\Program Files\PST\Portrait Foundation\Common\Bin". For each DLL do the following:

- Add a new row to the Component table. Use the name of the assembly for the new component and select the DIR_PST_PORTRAIT_FOUNDATION_COMMON_BIN directory. Set the Attributes to "0x0008" and the KeyPath to the name of the assembly.

- Add a new row to the File table for the assembly and select the component created previously. If there is a PDB file this can also be added to the same component.

- Map an entry in the FeatureComponents table, set the feature to "ProcessServerComponents" and select the new component.

Some of these files need to be registered. To find out what type of component each file is look in the following two folders:

- \output\Repository\CRMServer\CSharp\register

- \output\Repository\CRMServer\VC\register

For the files in these folders perform to relevant additional step...

CSharp register (COM visible .NET assembly)

Switch to the Components tab view and select the component for the file that needs to be registered. Right click and select "Import .reg file" then navigate to the registry file for that component and click Open. This will import all the relevant settings to the Registry table.

VC register (COM DLL)

Add an entry in the SelfReg table for the file that needs to be registered.

If the SelfReg table is not visible in **InstEd**, select Tables > Add Predefined Tables..., scroll to the bottom of the list, select "SelfReg" and click OK.

For more details on this MSI table see: http://msdn.microsoft.com/en-us/library/windows/desktop/aa371608(v=VS.85).aspx

Other files

Any other files such as the those referenced in files.txt by the old Non MSI Implementation Install must be added manually. It is recommended that a separate batch file is created to copy these files into the desired installation location under "**\Transform\Program Files\PST\Portrait Foundation**".

4.3.3.4 Troubleshooting

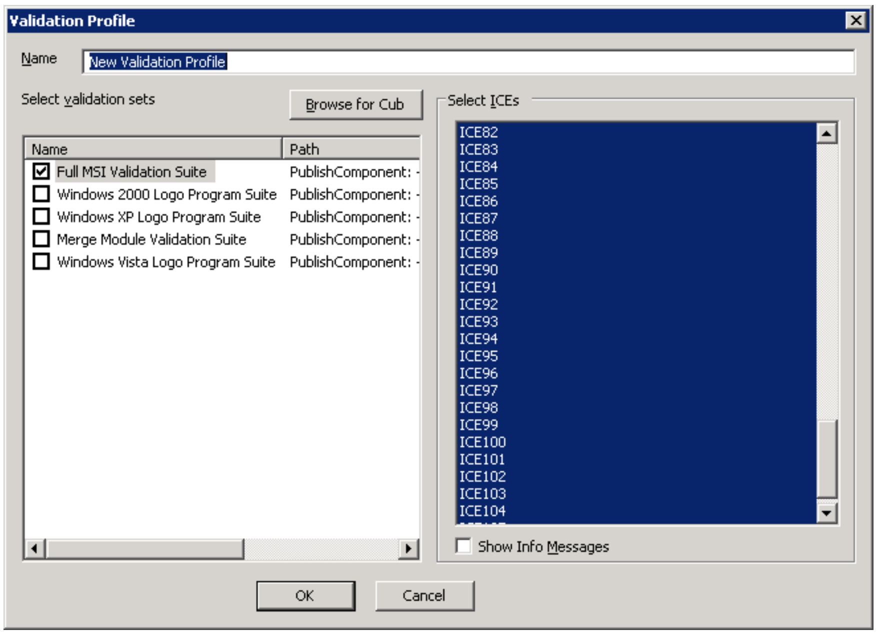
Sometimes the transform will cause the install to become invalid. In this case save the mst with a different name and try the new file instead. If this works, simply delete the old mst and rename the new one.

Validation

To avoid any possible install problems it is recommended that the MST is validated. This can be done using **InstEd**.

Select Tables > Validate

Click "..." to select the following Validation Profile and click OK.


Validation Profile
Name New Validation Profile
Select validation sets
Browse for Cub
Select ICEs
Name
Path
Full MSI Validation Suite PublishComponent: -
Windows 2000 Logo Program Suite PublishComponent: -
Windows XP Logo Program Suite PublishComponent: -
Merge Module Validation Suite PublishComponent: -
Windows Vista Logo Program Suite PublishComponent: -
ICE82
ICE83
ICE84
ICE85
ICE86
ICE87
ICE88
ICE89
ICE90
ICE91
ICE92
ICE93
ICE94
ICE95
ICE96
ICE97
ICE98
ICE99
ICE100
ICE101
ICE102
ICE103
ICE104
Show Info Messages
OK
Cancel

Click OK again to perform the validation.

Look for errors like:

```
ICE21   ERROR   Component: 'AIT_HRZ_Products.dll' does not belong to any Feature.
```

4.3.3.5 Applying the Transform to create an MSI

It is possible to apply an MSI Transform to an MSI using the command line.

```
msiexec /i "Portrait Implementation.msi" TRANSFORMS="All Applications.mst"
```

But we also supply a batch file (Apply_Transform.bat) that will roll these two files together to create a new MSI. It also updates the ProductVersion in the MSI to match the value in ait_version.cfg and will Sign the file with the Digital Certificate if one was supplied when setting up the Build Environment.

Simply navigate to the Transform folder from a command prompt and run the batch file passing in the name of the MSI Transform.

```
Apply_Transform.bat "All Applications"
```

This batch file will also output the final install collateral to the folder setup in the Build Environment as the target install location. Each version will have its own sub folder to represent the build number. This sub folder should simply contain the Implementation MSI install and the "**Program Files**" folder.

4.3.3.6 Running the MSI install

To run the install simply double click on the MSI file from the target install location in Windows Explorer.

Command line options

Being a basic MSI installer it is possible to supply additional parameters on the command line to default certain MSI properties. For a full list of Properties in any MSI, open it in a tool line **InstEd** or **Orca** and look at the uppercase entries in the Property table. The relevant properties for the "Portrait Implementation.msi" shown in section 4.3.1 are:

- SYSTEMNAME
- WEBSITE
- PORTNUMBER

- USERDOMAIN
- USERNAME
- PASSWORD
- PASSWORDCONFIRM

Your MSI install may contain others, for example the "All Applications.msi" also contains a TELEPHONY property with three possible values NONE, DEMO or FULL.

So to run the MSI Implementation install silently use the following command:

```
msiexec /i "All Applications.msi" USERDOMAIN=<domain> USERNAME=<username>
PASSWORD=<password> PASSWORDCONFIRM=<password> /qn
```

For further details on **MSIEXEC** please refer to:

http://technet.microsoft.com/en-us/library/cc759262(WS.10).aspx

4.3.3.7 Multi-tenant installs

The Implementation install has the built-in capability of having 4 instances installed. By default the first instance is installed by double clicking on the MSI or using the following command:

```
msiexec /i "All Applications.msi"
```

To setup other Portrait Foundation systems using the Implementation install, the MSI must be launched via the command line:

```
msiexec /i "All Applications.msi" MSINEWINSTANCE=1 TRANSFORMS=":InstanceId1.mst"
msiexec /i "All Applications.msi" MSINEWINSTANCE=1 TRANSFORMS=":InstanceId2.mst"
msiexec /i "All Applications.msi" MSINEWINSTANCE=1 TRANSFORMS=":InstanceId3.mst"
```

The MSINEWINSTANCE parameter tells the MSI that we are installing a new instance and the TRANSFORMS parameter (e.g. =":InstanceId2.mst") indicates which instance to use. These transforms are embedded within the MSI file and are denoted by the use of the colon in the instance name. Each instances will have its own entry in "Programs and Features".

Be sure to select a different Portrait Foundation system for each instance.

For more details on multiple instance installs please refer to:

http://msdn.microsoft.com/en-us/library/windows/desktop/aa369528(v=vs.85).aspx

Appendix A Quick Reference

The following documents provide additional details of the extensibility points described in this previous sections.

1. Extending Applications using .NET *(Portrait NET SDK)*
2. ASP.NET Application Development Guide *(NET ADK User Guide)*
3. Localizing ASP.NET Applications
4. Host Integration Framework User Guide
5. Portrait Integration White Paper
6. Portrait Data Mart Creation Utility User Guide
7. Integrating Telephony
8. Integrating Simplex Channels
9. Batch Load Framework Configuration Guide
10. Batch Load Framework Operational Guide
11. Foundation Reporting
12. Telephony client integration

The table below lists all of the areas in which it is possible to extend Portrait Foundation functionality through development activities. For areas where SDK support is not available please contact your Portrait Support helpdesk if you have a specific need for this functionality.

Area of extensibility	SDK	Doc Reference	Contact Support
Integration with external line-of-business systems	✓	1, 4, 5	
Additional Nodes for use in Process Models	✓	1	
Integration with inbound document sources	✓	1	
Use of different servers for sending emails	✓	1	
Accessing different document management systems	✓	1	
Use of different encryption algorithms in Data Comparison Strategies	✓	1	
Creation of new ASP .NET pages within Portrait Foundation applications	✓	2	
Localisation of application content		3	
Integration of new telephony systems		7, 12	✓
Integration with third party printing products		8	✓
Loading bulk data into the Portrait Foundation database		9, 10	
Extending the default Data Mart		6	
Reporting on operational data in the Foundation database		11	