IVI-3.17: Installation Requirements Specification

December 19, 2022 Edition
Revision 2.8

Important Information

IVI-3.17: Installation Requirements Specification is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at www.ivifoundation.org.

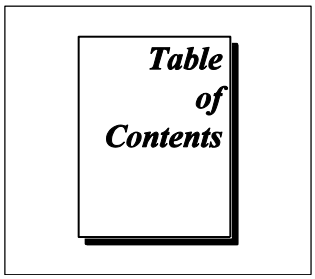
Warranty

The IVI Foundation and its member companies make no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The IVI Foundation and its member companies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Trademarks

Product and company names listed are trademarks or trade names of their respective companies.

No investigation has been made of common-law trademark rights in any work.



IVI-3.17: Installation Requirements Specification

IVI Installation Requirements Revision History

This section is an overview of the revision history of the IVI Installation Requirements specification.

Table 1-1. IVI Installation Requirements Specification Revisions

Revision Number	Date of Revision	Revision Notes
Revision 1.0	March 30, 2009	Original release. Created from the Section 6, *Installation Requirements*, in *IVI-3.1: Driver Architecture Specification*.
Revision 1.1	February 19, 2010	Editorial changes to add Windows 7 as a supported OS.
Revision 2.0	June 9, 2010	Added .NET architecture
Revision 2.0	June 30, 2011	Editorial change: Added text to permit placing some files in locations determined by the driver vendor.
Revision 2.1	January 13, 2012	Minor change: Added Section 6.4. Modified Sections 7.1.1 and 7.1.2.
Revision 2.2	March 6, 2013	Minor changes to add Windows 8 as a supported OS.
Revision 2.3	Dec. 3, 2013	Minor change: Change the IVI.NET Component Version-Specific Directory name to use <FullVersion> rather than <MajorMinorVersion>. Change all descriptive text to reflect this. Remove all text that describes .NET installation behavior peculiar to changing only the Build revision number. Remove the definition of <MajorMinorVersion>.
Revision 2.3	May 9, 2014	Editorial change: Added text in Section 4.2.3 to more explicitly permit vendors to point AssemblyFoldersEx registry keys to assemblies in vendor-specific locations and to suggest that vendors may register older versions of assemblies.
Revision 2.3	September 5, 2014	Editorial change: In Sections 6.3.1 and 6.3.2, added a requirement that the user must accept the IVI Foundation License in the IVI Shared Components and IVI.NET Shared Components installers to proceed with installation.
Revision 2.3	December 10, 2014	Editorial changes: In sections 2.3.1.1.2 and 2.3.2.1.2, added text to allow bundling the 32-bit and 64-bit installers into one .exe file; in Section 2.3.2, recommended side-by-side installation; in Section 2.5.2.1 removed parantheses from around FullVersion in Component Version-Specific Directories; in Section 4.2.3, Table 4-5, added parentheses around FwkVerShortName in the name of the registry key under AssemblyFoldersEx and distinguish between driver installers and IVI.NET shared component installer; in Section 5.2, step 7, added a sentence about policy file.
Revision 2.4	February 11, 2015	Minor change: In Section 6.4.1, addressed naming and availability of the IVI shared component .exe installer files and .msi installer files.

Table 1-1. IVI Installation Requirements Specification Revisions

Revision 2.4	August 6, 2015	Editorial changes to remove Windows 2000 and add Windows 10 as supported operating systems
Revision 2.5	June 7, 2016	Minor change to remove support for Windows XP and Windows Vista
Revision 2.6	November 03, 2016	In Section 4.4, reversed the policy of unlocking the IVI directory to a policy of locking the IVI directory in the IVI Shared Component installer.
Revision 2.7	November 18, 2016	Minor change to add a directory for 4.6 to the .NET Framework Version Directories and to clarify where to install drivers that require a patch version of the .NET Framework.
Revision 2.7	October 19, 2018	Editorial change to the 4.2.3 to change <CLRVersion> to <TargetFrameworkVersion>.
Revision 2.8	June 29, 2021	Minor change to describe file names and behavior of new installers.
Revision 2.8	December 19, 2022	Editorial change to add Windows 11 as a supported operating system.

1. Overview of the IVI Installation Requirements Specification

1.1 Introduction

This section specifies the required and optional features for the installation programs that install IVI drivers and IVI shared components on user systems. This section identifies the features for which the IVI Foundation allows installation programs flexibility in implementation.

1.2 Definition of Installation Terms

The following are some commonly used installation terms within this section.

Dialog Mode Installation

The default user interface mode of installation. Users interact with the installer to set installation options. The installer displays status information to the user.

Silent Mode Installation

A user interface mode of installation where the user does not interact with the installer user interface to set installer options. Instead, installer options are set on the command-line.

MSI

Microsoft Installer for Windows. A technology developed by Microsoft for installing software components on Windows operating systems.

User Account Control (UAC)

A Windows 7 and later feature that manages user and application security privileges.

Standard Privileges

The default UAC privileges on Windows 7 and later for all applications.

Admin Privileges

The elevated (full) UAC privileges on Windows 7 and later that allow applications to perform administrative tasks.

Global Assembly Cache (GAC)

The cache used to store and list public/shared .NET assemblies available on the system.

1.3 Definition of IVI Installation Terms

The following are commonly used IVI installation terms within this section.

IVI Installer

An installation program that implements the requirements specified by the IVI Foundation for the purpose of installing IVI drivers or IVI shared components.

IVI Driver Installer

An IVI installer that installs IVI driver files.

IVI Standard Directory Tree

The directory tree into which driver installations place all files, except for files that must be in directories specific to ADEs and files installed to the GAC. The IVI shared components are also installed in the IVI standard directory tree.

IVI Standard Root Directory <IVIStandardRootDir>

The root of the IVI standard directory tree for all driver installations and shared component installations.

Windows 7 (32-bit), Windows 8 (32-bit), and Windows 10 (32-bit) have only a 32-bit IVI standard root directory. Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11 have both a 32-bit IVI standard root directory and a 64-bit IVI standard root directory.

In this specification, <IVIStandardRootDir> refers to the 32-bit IVI standard root directory or the 64-bit IVI standard root directory, depending on whether the application, driver, or installer is 32-bit or 64-bit. The term <IVIStandardRootDir32> refers to the 32-bit IVI standard root directory and the term <IVIStandardRootDir64> refers to the 64-bit IVI standard root directory.

The default IVI standard root directory is <ProgramFilesDir>\IVI Foundation\IVI. Refer to Section 0,4.2

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on <ProgramFilesDir>.

IVI Data Directory <IVIDataDir>

The IVI data directory contains the master IVI configuration store. The IVI data directory does not contain driver specific or vendor specific files. The default IVI data directory is <ProgramDataDir>\IVI Foundation\IVI. Refer to Section 0,4.2
A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.
Determining System Directories and Registry Keys, for information on <ProgramDataDir>.

Full Version <FullVersion>

The Full Version defines the version of the product in <Major.Minor.Build> format.

The `Build` field indicates the patch level.

Full Version Compressed `<FullVersionCompressed>`

The Full Version Compressed defines the version of the product in `<MajorMinorBuild>` format, with no dots.

The `Build` field indicates the patch level.

1.3.1 Definition of IVI-COM/IVI-C Installation terms

IVI-COM/IVI-C Shared Components

The set of shared components that IVI-COM and IVI-C drivers use.

IVI-COM/IVI-C Installer

An installation program that implements the requirements specified by the IVI Foundation for the purpose of installing IVI-C drivers, IVI-COM drivers or IVI-COM/IVI-C shared components.

IVI-COM/IVI-C Driver Installer

An IVI installer that installs IVI-C and/or IVI-COM driver files.

IVI-COM/IVI-C Shared Component Cleanup Utility

A utility created and distributed by the IVI Foundation that removes IVI-COM/IVI-C shared component files and registry entries from a system.

IVI-COM/IVI-C Standard Directory Tree

The directory tree into which IVI-COM and IVI-C driver installations place all files, except for files that must be in directories specific to ADEs and files installed into the GAC. The IVI-COM/IVI-C shared components are also installed in the IVI standard directory tree.

IVI-COM/IVI-C Shared Component Installer

An installer developed and distributed by the IVI Foundation that creates the directory tree for IVI-COM/IVI-C shared component files and driver files, installs all the IVI-COM/IVI-C shared component files, and creates the directory and registry entries for the master IVI configuration store.

The term 32-bit IVI-COM/IVI-C Shared Component Installer refers to the IVI-COM/IVI-C shared component installer that installs on a 32-bit operating system, and the term 64-bit IVI-COM/IVI-C Shared Component Installer refers to the IVI-COM/IVI-C shared component installer that installs on a 64-bit operating system.

Standard IVI-COM/IVI-C Driver Specific Directory

The directory into which an IVI-COM/IVI-C driver installer places files that it does not disperse to the standard common files directories or a standard directory of an ADE. The standard driver specific directory is `<IVIStandardRootDir>\Drivers\<ComponentIdentifier>` for IVI-COM drivers or `<IVIStandardRootDir>\Drivers\<Prefix>` for IVI-C drivers or for IVI-COM drivers that are packaged with C wrappers.

IVI-COM/IVI-C Shared Component Directory

The directory that the IVI Foundation specifies to contain the shared component files that are not installed into the standard common files directories. The IVI-COM/IVI-C shared component directory is `<IVIStandardRootDir>\Components`.

Dispersed File

An IVI-COM or IVI-C driver file or shared component file that is installed into one of the standard common files directories or into a standard directory of the operating system, an ADE, or another application program.

Non-dispersed File

An IVI-COM or IVI-C driver file or shared component file that is not installed into one of the standard common files directories or into a standard directory of the operating system, an ADE, or another application program. A non-dispersed driver file is installed into the standard driver specific directory for the driver. A non-dispersed shared component file is installed into the IVI shared component directory tree. Examples of non-dispersed files include help documentation, source code, and compliance documents.

Standard Common Files Directories

The directories that the IVI Foundation specifies to contain certain common types of files, such as DLLs, import libraries, and include files. It is useful to place these common types of files into separate directories so that ADEs can find them. The standard common files directories are in the IVI standard root directory and contain IVI-COM/IVI-C shared component files as well as IVI-COM and IVI-C driver files.

Note: An installer *disperses* files when it installs them into the standard common files directories or into a standard directory of the operating system, an ADE, or another application program.

1.3.2 Definition of IVI.NET Installation Terms

IVI.NET Shared Components

The set of shared components thatIVI.NET drivers use.

IVI.NET Installer

An installation program that implements the requirements specified by the IVI Foundation for the purpose of installing IVI.NET drivers or IVI.NET shared components.

IVI.NET Driver Installer

An IVI.NET installer that installs IVI.NET driver files. Because IVI.NET driver files install into a different part of the IVI directory hierarchy and multiple versions of the same driver may install side-by-side, the requirements for IVI.NET drivers are different than the requirements for IVI-COM and IVI-C drivers.

IVI.NET Shared Component Installer

An IVI.NET installer created and distributed by the IVI Foundation that installs all the IVI.NET shared component files. Because the IVI.NET shared component files install into a different part of the IVI directory hierarchy and multiple versions of the shared components may install side-by-side, the requirements for installing IVI.NET shared components are different than the requirements for installing IVI-COM/IVI-C shared components.

The IVI.NET shared component installer does not create the standard IVI root directory or the directory and registry entries for the master IVI configuration store. Thus, neither the IVI.NET shared component installer nor IVI.NET driver installers work unless the IVI-COM/IVI-C shared component installer has been successfully executed on the machine.

The term 32-bit IVI.NET Shared Component Installer refers to the IVI.NET shared component installer that installs on a 32-bit operating system, and the term 64-bit IVI.NET Shared Component Installer to refers to the IVI.NET shared component installer that installs on a 64-bit operating system.

IVI.NET Standard Directory Tree

The directory tree into which IVI.NET driver installations place all files, except for files that must be in directories specific to ADEs and files installed into the GAC. The IVI.NET shared components are also installed in the IVI.NET standard directory tree.

IVI.NET Standard Root Directory <IviNetStandardRootDir>

The root of the IVI.NET standard directory tree for all IVI.NET driver and shared component installations.

Windows 7 (32-bit), Windows 8 (32-bit), and Windows 10 (32-bit) have only a 32-bit IVI.NET standard root directory. Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11 have both a 32-bit IVI.NET standard root directory and a 64-bit IVI.NET standard root directory.

In this specification, <IviNetStandardRootDir> refers to the 32-bit IVI.NET standard root directory or the 64-bit IVI.NET standard root directory, depending upon whether the application, driver, or installer is 32-bit or 64-bit. The term <IviNetStandardRootDir32> refers to the 32-bit IVI.NET standard root directory and the term <IviNetStandardRootDir64> refers to the 64-bit IVI.NET standard root directory.

The IVI.NET standard root directory is <IVIStandardRootDir>\Microsoft.NET.

IVI.NET Framework Platform Directory <IVINetFrameworkPlatformDir>

The IVI.NET framework platform directory provides the location for all IVI.NET drivers and shared components that are specific to one of the following: 32-bit support or 64-bit support.

Windows 7 (32-bit), Windows 8 (32-bit), and Windows 10 (32-bit) have only a 32-bit IVI.NET framework platform directory. Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11 have both a 32-bit IVI.NET framework platform directory and a 64-bit IVI.NET framework platform directory.

In this specification, <IVINetFrameworkPlatformDir> refers to the 32-bit IVI.NET framework platform directory or the 64-bit IVI.NET framework platform directory, depending upon the whether the application, driver, or installer is 32-bit or 64-bit. The term <IVINetFrameworkPlatformDir32> refers to the 32-bit

IVI.NET framework platform directory and the term <IVINetFrameworkPlatformDir64> refers to the 64-bit IVI.NET framework platform directory.

The 32-bit IVI.NET framework platform directory is <IviNetStandardRootDir32>\Framework. The 64-bit IVI.NET framework platform directory is <IviNetStandardRootDir64>\Framework64.

Framework-Dependent Variant

A version of an IVI.NET driver (or a version of the IVI.NET shared components) as considered in relation to the minimum major.minor version of the .NET Framework that it requires.

.NET Framework Version Directories <FrameworkVersionDir>

The .NET Framework version directories provide separate locations for IVI.NET drivers and shared components that require different minimum major.minor versions of the .NET Framework. A different .NET Framework Version directory exists for each unique major.minor version of the .NET Framework, starting with .NET Framework 2.0. IVI.NET drivers that require a patch version of the .NET Framework, for example 4.6.1, are installed to the corresponding major.minor folder, for example v4.6.

Prior to .NET 4.6, the value of <FrameworkVersionDir> matches the .NET Framework directory name under %Windows%\Microsoft.NET\Framework[64]. For .NET 4.6 and later, the value of <FrameworkVersionDir> is v<framework major version>.<framework minor version>.

The list of available frameworks and their associated values at the time of this specification are as follows:

Table 1-2. .NET Framework Version Directories

.NET Framework Major.Minor Version	<FrameworkVersionDir>
2.0	v2.0.50727
3.0	v3.0
3.5	v3.5
4.0	v4.0.30319
4.5	v4.5.50709
4.6	v4.6

.NET Framework Version Short Name <FwkVerShortName>

The .NET Framework version short name is used to provide .NET Framework version-specific names for registry keys, Software Module Table entries, and any Start Menu folders. Refer to Section 4.3.1, *Target .NET Framework Versions*, in *IVI-3.1: Driver Architecture Information*, for specific valid values of the .NET Framework version short name.

IVI.NET Driver Namespace <DriverNamespace>

The IVI.NET driver namespace uniquely identifies the instrument driver. Refer to Section 5.17.1, *IVI.NET Namespaces*, in *IVI-3.1: Driver Architecture Information*, for the format of IVI.NET instrument driver namespaces.

IVI.NET Component Version-Specific Directory

The directory into which an IVI.NET driver installer or the IVI.NET shared component installer is permitted to install files under the IVI.NET Standard Root directory.

For instrument drivers, the IVI.NET Component Version-Specific Directory is `<IVINetFrameworkPlatformDir>\<FrameworkVersionDir>\<DriverNamespace> <FullVersion>`.

For IVI.NET Shared Components, the IVI.NET Component Version-Specific Directory is `<IVINetFrameworkPlatformDir>\<FrameworkVersionDir>\IVIFoundationSharedComponents <FullVersion>`.

Vendors may install driver support files, such as examples, to other locations that they deem appropriate.

IVI.NET Version Identification File

The file installed by an IVI.NET driver installer or IVI.NET shared component installer to identify the version of the driver or shared components.

For instrument drivers, the IVI.NET version identification file name shall be `<DriverNamespace>Version.dll`.

For IVI.NET Shared Components, the IVI.NET version identification file name shall be `IVIFoundationSharedComponentsVersion.dll`.

Design-Time Components

Components that are used to develop and compile applications. These components include assemblies, IntelliSense help files, and online help files. Design-time assemblies are referenced by user project files and utilized during compilation. Design-time components are installed to the appropriate IVI.NET Component Version-Specific Directory.

Managed Run-Time Components

.NET assemblies and policy files that are utilized by user applications at run time. Managed run-time components are installed to the GAC.

Unmanaged Run-Time Components

Files, other than .NET assemblies and policy files, that are utilized by user applications at run time.

Patching

The process of updating existing IVI.NET drivers or IVI.NET shared components by modifying the existing version rather than installing another version side-by-side. Patching is done to fix bugs in a way such that existing applications do not have to be rebuilt.

Note: Patching shall not be used when .NET Interface types change.

1.3.3 Definition of IVI Driver Installer Bitness Types

To avoid confusion, this specification uses a naming convention for the various types of IVI driver installers as they relate to different driver and operating system bitnesses.
The format for the names is:

[singular | unified] <*driver bitness*> driver installer [<*supported operating system bitness*>]

- "singular" denotes that the installer installs a driver of only one bitness
- "unified" denotes that the installer installs both 32-bit and 64-bit variants of a driver.
- <*driver bitness*> *can* take the form of "32-bit", "64-bit", or "32-bit/64-bit".
- <*supported operating system bitness*> is used when an installer type can support operating systems of different bitnesses and it is necessary to specify one or more particular operating system bitness(es). It can take one of the following values "(32-bit OS)", "(64-bit OS)", or "(32-bit/64-bit OS)".

This specification uses the following set of IVI driver installer bitness types.

Singular 32-Bit Driver Installer

A *singular 32-bit driver installer* installs only a 32-bit version of a driver.

A *singular 32-bit driver installer (32-bit OS)* installs a 32-bit version of a driver on 32-bit operating systems. This installer refuses to install on 64-bit operating systems.

A *singular 32-bit driver installer (64-bit OS)* installs a 32-bit version of a driver on 64-bit operating systems. This installer refuses to install on 32-bit operating systems.

A *singular 32-bit driver installer (32-bit/64-bit OS)* installs a 32-bit version of a driver on both 32-bit and 64-bit operating systems.

Singular 64-bit Driver Installer

A *singular 64-bit driver installer* installs only a 64-bit version of a driver on 64-bit operating systems. This installer refuses to install on 32-bit operating systems.

Unified 32-bit/64-bit Driver Installer

A *unified 32-bit/64-bit driver installer* installs both a 32-bit version and a 64-bit version of a driver on 64-bit operating systems. This installer always installs either both or neither of the drivers. This installer refuses to install on 32-bit operating systems.

2. Features and Intended Use of Installers

2.1 Introduction

This section describes the features and intended use of IVI installers. It provides an overview of the installation directories and types of installers.

2.2 Installers

The IVI Foundation specifies installation requirements for two types of installation programs: an IVI driver installer and an IVI shared component installer. Driver developers are responsible for packaging and distribution of the IVI drivers that they create. The IVI Foundation provides an installer for all the IVI shared components. IVI driver suppliers may distribute the IVI shared component installer. They can do so by calling the IVI shared component installer from the IVI driver installer or by distributing the IVI shared component installer along with the IVI driver installer.

2.3 IVI Driver Installation

2.3.1 IVI-COM/IVI-C Driver Installation

An IVI-COM/IVI-C driver installation program installs the driver files to standard directories, creates Windows registry entries for the IVI-C or IVI-COM driver, registers the driver with the IVI configuration store, and registers an uninstaller for the driver. All IVI-COM/IVI-C driver installations are made within a root directory that the user specifies when installing an IVI-COM or IVI-C driver for the first time. The IVI-COM/IVI-C shared components are also installed within the same root directory. The IVI-COM or IVI-C driver is immediately usable after installation. Multiple versions of an installed IVI-COM or IVI-C driver cannot co-exist on the same machine.

2.3.1.1 IVI-COM/IVI-C Driver Installers and Bitness

IVI-COM/IVI-C driver installers may install 32-bit drivers, 64-bit drivers, or both. IVI-COM/IVI-C driver installers may run on 32-bit operating systems, 64-bit operating systems, or both.

2.3.1.1.1 Valid Uses of Driver Installer Bitness Types for IVI-COM/IVI-C Driver Installers

This section specifies the IVI driver installer bitness types that are valid for IVI-COM/IVI-C driver installers, based on the operating systems a driver supplier supports and the bitness of the IVI-COM or IVI-C drivers the supplier provides. For the definition of the types, refer to Section 1.3.3, *Definition of IVI Driver Installer Bitness Types*.

Table 2-1. Valid installer bitness types when only a 32-bit driver is available

Supported Operating Systems	**Valid Combinations of Driver Installation Program Types**
Only 32-bit operating systems	Singular 32-bit driver installer (32-bit OS)
Only 64-bit operating systems	Singular 32-bit driver installer (64-bit OS)
Both 32-bit and 64-bit operating systems	**Option A:** Singular 32-bit driver installer (32-bit OS) AND Singular 32-bit driver installer (64-bit OS)

	Note: This option is recommended for drivers that call the IVI shared component installer or are transitioning to 64-bit driver support. **Option B:** Singular 32-bit driver installer (32-bit OS/64-bit OS) Note: This option is recommended for drivers that do not call the IVI-COM/IVI-C shared component installer. If the installer calls the IVI-COM/IVI-C shared component installer, then this option requires either calling the legacy IVI-COM/IVI-C shared component installer or calling both the 32-bit and 64-bit IVI shared component installers.

Table 2-2. Valid installer bitness types when only a 64-bit driver is available

Supported Operating Systems	Valid Combinations of Driver Installation Program Types
Only 64-bit operating systems	Singular 64-bit driver installer

Table 2-3. Valid installer bitness types when both a 32-bit and a 64-bit driver are available

Supported Operating Systems	Valid Combinations of Driver Installation Program Types
32-bit driver is supported on both 32-bit and 64-bit operating systems 64-bit driver is supported on 64-bit operating systems	Singular 32-bit driver installer (32-bit OS) AND Unified 32-bit/64-bit driver installer (64-bit OS) Note: This option ensures that the driver revision for both the 32-bit version and 64-bit version of the driver on a 64-bit operating system are the same.
32-bit driver is supported only on 32-bit operating systems 64-bit driver is supported on 64-bit operating systems Note: This excludes installation of a 32-bit driver on a 64-bit operating system	Singular 32-bit driver installer (32-bit OS) AND Singular 64-bit driver installer (64-bit OS) Note: The Singular 32-bit driver installer (32-bit OS) shall refuse to install on 64-bit operating systems.
32 bit driver is supported only on 64-bit operating systems 64-bit driver is supported on 64-bit operating systems	Unified 32-bit/64-bit driver installer (64-bit OS) Note: This ensures that the driver revision for both the 32-bit version and 64-bit version of the driver on a 64-bit operating system are the same.

2.3.1.1.2 Recommended IVI-COM/IVI-C Driver Installer Approach

The IVI Foundation recommends that driver suppliers build a 32-bit driver that works on both 32-bit and 64-bit operating systems and a 64-bit driver that works on 64-bit operating systems. The IVI Foundation recommends that driver suppliers distribute these drivers using the following installer bitness types:

- Singular 32-bit driver installer (32-bit OS)
- Unified 32-bit/64-bit driver installer (64-bit OS)

Note: The Singular 32-bit driver installer and Unified 32-bit/64-bit driver installer may be bundled into a single .exe installer that can run on both a 32-bit OS and a 64-bit OS.

Prior to version 2.0 of this specification, IVI-COM and IVI-C drivers were only 32-bit and shipped with installers packaged as singular 32-bit driver installers (32-bit OS/64-bit OS). Driver suppliers may continue to distribute singular 32-bit driver installers (32-bit OS/64-bit OS), but *only if* they do not also distribute 64-bit versions of the same drivers.

Changing an existing driver that ships with a singular 32-bit driver installer (32-bit OS/64-bit OS) to the recommended approach requires modifying the build process for the driver and the driver installer. Driver suppliers changing to the recommended approach should pay particular attention to the following:

- The 32-bit driver installer must be modified to refuse to install on 64-bit operating systems.
- The 64-bit import libraries for IVI-C drivers must be added to the 32-bit driver installer.
- The 32-bit and 64-bit driver DLLs must have the same MajorVersion, MinorVersion, and BuildVersion.
- The DLL FileVersion must be updated with at least an incremented BuildVersion as compared to the version before adding the 64-bit driver.

Note: 64-bit IVI-C installers must also set the ModulePath64 property in the IVI configuration store.

2.3.2 IVI.NET Driver Installation

An IVI.NET driver installation program installs the driver files to standard directories, creates Windows registry entries for the IVI.NET driver, registers the driver with the IVI configuration store, and registers an uninstaller for the driver. The IVI.NET driver is immediately usable after installation. Multiple versions of an installed IVI.NET driver can coexist on the same machine. The IVI Foundation strongly recommends that IVI.NET driver installers install side-by-side with other versions of the same driver, that is, without uninstalling the other versions of the driver on the system.

2.3.2.1 IVI.NET Driver Installers and Bitness

IVI.NET driver installers may install 32-bit drivers, 64-bit drivers, or both. IVI.NET driver installers may run on 32-bit operating systems, 64-bit operating systems, or both.

2.3.2.1.1 Valid Uses of Driver Installer Bitness Types for IVI.NET Driver Installers

This section specifies the IVI driver installer bitness types that are valid for IVI.NET driver installers, based on the operating systems a driver supplier supports and the bitness of the IVI.NET drivers the supplier provides. For the definition of the types, refer to Section 1.3.3, *Definition of IVI Driver Installer Bitness Types*.

Table 2-4. Valid installer bitness types when only a 32-bit driver is available

Supported Operating Systems	Valid Combinations of Driver Installation Program Types
Only 32-bit operating systems	Singular 32-bit driver installer (32-bit OS)
Only 64-bit operating systems	Singular 32-bit driver installer (64-bit OS)
Both 32-bit and 64-bit operating systems	**Option A:** Singular 32-bit driver installer (32-bit OS) AND Singular 32-bit driver installer (64-bit OS) Note: This option is recommended for drivers that call the IVI.NET shared component installer or are transitioning to 64-bit driver support.
	Option B: Singular 32-bit driver installer (32-bit OS/64-bit OS) Note: This option is recommended for drivers that do not call the IVI.NET shared component installer. If the installer calls the IVI.NET shared component installer, then this option requires calling both the 32-bit and 64-bit IVI.NET shared component installers.

Table 2-5. Valid installer bitness types when only a 64-bit driver is available

Supported Operating Systems	Valid Combinations of Driver Installation Program Types
Only 64-bit operating systems	Singular 64-bit driver installer

Table 2-6. Valid installer bitness types when both a 32-bit and a 64-bit driver are available

Supported Operating Systems	Valid Combinations of Driver Installation Program Types
32-bit driver is supported on both 32-bit and 64-bit operating systems 64-bit driver is supported on 64-bit operating systems	Singular 32-bit driver installer (32-bit OS) AND Unified 32-bit/64-bit driver installer (64-bit OS) Note: This option ensures that the driver full version for both the 32-bit version and 64-bit version of the driver on a 64-bit operating system are the same.
32-bit driver is supported only on 32-bit operating systems 64-bit driver is supported on 64-bit operating systems Note: This excludes installation of a 32-bit driver on a 64-bit operating system	Singular 32-bit driver installer (32-bit OS) AND Singular 64-bit driver installer (64-bit OS) Note: The Singular 32-bit driver installer (32-bit OS) shall refuse to install on 64-bit operating systems.

32 bit driver is supported only on 64-bit operating systems 64-bit driver is supported on 64-bit operating systems	Unified 32-bit/64-bit driver installer (64-bit OS) Note: This ensures that the driver full version for both the 32-bit version and 64-bit version of the driver on a 64-bit operating system are the same.

2.3.2.1.2 Recommended IVI.NET Driver Installer Approach

The IVI Foundation recommends that driver suppliers build a 32-bit driver that works on both 32-bit and 64-bit operating systems and a 64-bit driver that works on 64-bit operating systems. The IVI Foundation recommends that driver suppliers distribute these drivers using the following installer bitness types:

- Singular 32-bit driver installer (32-bit OS)
- Unified 32-bit/64-bit driver installer (64-bit OS)

Note: The Singular 32-bit driver installer and Unified 32-bit/64-bit driver installer may be bundled into a single .exe installer that can run on both a 32-bit OS and a 64-bit OS.

2.3.2.1.3 IVI.NET Driver Installers and .NET Framework Versions

An IVI.NET driver supplier might want to take advantage of features in a .NET Framework version that are not available in prior .NET Framework versions that the driver supports. To do this, the driver supplier creates two different framework-dependent variants of the driver. In such a case, each variant shall be installed in the .NET Framework Version directory for the minimum major.minor .NET Framework version that the variant requires. A single IVI.NET driver installer shall install exactly one framework-dependent variant.

2.3.2.1.4 IVI.NET Driver Installers and Design-Time Support

The default behavior of an IVI.NET driver installer shall be to install all design-time and run-time components. An IVI.NET driver installer may support conditional installation of some or all design-time components.

2.4 IVI Shared Component Installation

The IVI Foundation made significant packaging changes to the shared components installers in 2021. For the most part, these changes were either cosmetic or invisible to users. The biggest change was to combine the previous 32-bit and 64-bit installers into a single installer that can run on both 32-bit and 64-bit Windows.

2.4.1 IVI-COM/IVI-C Shared Component Installation

The IVI Foundation provides a single IVI-COM/IVI-C shared components installation program, which installs 32-bit IVI-COM/IVI-C shared components on 32-bit operating systems and both 32-bit and 64-bit IVI-COM/IVI-C shared components on 64-bit operating systems.

The IVI-COM/IVI-C shared component installer program installs the following IVI-COM/IVI-C shared components:

- IVI-C, IVI-COM, and IVI.NET Configuration Servers
- IVI Floating Point Services
- IVI-C Shared Components

- IVI-COM Session Factory
- IVI Type Libraries
- IVI Primary Interop Assemblies
- IviLxiSync Components

The latest IVI-COM/IVI-C shared component installation program is also available for public download from the IVI Foundation web site, `www.ivifoundation.org`, and are not distributed as an IVI-COM/IVI-C shared component.

IVI-COM and IVI-C drivers cannot be installed until the user has successfully installed the IVI-COM/IVI-C shared components for the appropriate operating system. IVI-COM/IVI-C driver installers have the ability to detect the presence of the shared components and to verify that they are of a version sufficient for the driver. If the shared components are not detected or are not of a sufficient version, the IVI-COM/IVI-C driver installer may call into the IVI-COM/IVI-C shared component installer program, thus providing a seamless install experience for the end user. Alternatively, the IVI-COM/IVI-C driver installer may require the user to run the IVI-COM/IVI-C shared component installer as a separate step before installing the driver. In that case, either the driver supplier distributes the IVI-COM/IVI-C shared component installer with the IVI-COM/IVI-C driver installer, or the IVI-COM/IVI-C driver installer directs the user to the IVI Foundation web site.

The IVI-COM/IVI-C shared component installation program checks for the presence of the shared components on the system. If the shared components are already present, the installer does not install files unless the shared components it contains have a version that is greater than or equal to the version of the shared components on the system.

To remove the IVI-COM/IVI-C shared components from a system, the user uses the standard Windows Control Panel facility to add and remove programs.

The IVI Foundation does not support or recommend the maintenance of multiple versions of the IVI-COM/IVI-C shared components on a system.

2.4.1.1 C/COM Legacy Installers

Legacy Installers (2008-2020)

Installers from 2008-2020 include multiple .exe packages, one for 32-bit operating systems, and another for 64-bit operating systems. This approach was used for the following versions:

- IVI Shared Components: versions 2.0.0 – 2.6.1.

See section 6.4, *IVI Shared Component Installer Files*, for details.

Note: The purpose of distributing the legacy installer is to allow suppliers to continue to distribute singular 32-bit driver installers (32-bit/64-bit OS) that call the IVI-COM/IVI-C shared component installer. If one of the two first two installers listed above has ever been run successfully, the legacy installer will do nothing.

32-bit Only Shared Components (2000-2008)

From its inception through 2008, The IVI Foundation only supported 32-bit C and COM components, although they could be installed on both 32-bit and 64-bit versions of Windows.

Installers from this time installed 32-bit IVI-COM/IVI-C shared components on both 32-bit and 64-bit operating systems. These legacy IVI-COM/IVI-C shared component installers have a version of 1.5.1 or less. The IVI Foundation will distribute version 1.5.1 of the legacy IVI-COM/IVI-C shared component installer, which complied with version 1.7 of IVI-3.1, *Driver Architecture Specification*, until the IVI Foundation determines that no IVI-COM/IVI-C driver installers require the legacy shared component installer. No maintenance updates will be made to the legacy IVI-COM/IVI-C shared component installer.

Note: The purpose of distributing the version 1.5.1 installer is to allow suppliers to continue to distribute singular 32-bit driver installers (32-bit/64-bit OS) that call the IVI-COM/IVI-C shared component installer. If a later installer has ever been run successfully, the legacy installer will do nothing.

2.4.2 IVI.NET Shared Component Installation

The IVI Foundation might want to take advantage of features in a .NET Framework version that are not available in prior .NET Framework versions that the IVI.NET shared components support. In this case, the IVI Foundation creates two framework-dependent variants of the same version of the IVI.NET shared components. Each variant shall be installed in the .NET Framework Version directory for the minimum major.minor .NET Framework version that the variant requires.

The IVI Foundation provides a single IVI.NET shared components installation program, which installs 32-bit IVI.NET shared components on 32-bit operating systems and both 32-bit and 64-bit IVI-COM/IVI-C shared components on 64-bit operating systems.

The IVI.NET shared component installer program installs the following IVI.NET shared components:

- IVI.NET Standard Inherent and Class Assemblies
- IVI.NET Standard Inherent and Class Assembly IntelliSense Files

All IVI.NET shared component assemblies shall be signed with the IVI Foundation public/private key pair, to allow installation to the GAC.

The latest IVI.NET shared component installation program is available for public download from the IVI Foundation web site, www.ivifoundation.org and is distributed separately from the IVI-COM/IVI-C shared component installation program.

The IVI Foundation supports side-by-side installations of multiple framework-dependent variants of multiple versions of the IVI.NET shared components on a system. Each IVI.NET shared component installation program checks for the presence on the system of the version and framework- dependent variant that it installs. The IVI.NET shared components installation program does not install files if the version and framework-dependent variant that it installs is already present on the system.

The IVI.NET shared components cannot be installed until the user has successfully installed the version of the .NET Framework that the shared components require. If the required version of the .NET Framework is not detected, the IVI.NET shared component installer exits and directs the user to run the .NET Framework installer.

The IVI.NET shared components cannot be installed until the user has successfully installed the IVI-COM/IVI-C shared components. The IVI.NET shared component installer attempts to detect the presence of the IVI-COM/IVI-C shared components and verify that they are of a version sufficient for the IVI.NET shared component installer. If the IVI-COM/IVI-C shared components are not detected or are not of a sufficient version, the IVI.NET shared component installer exits and directs the user to the IVI Foundation web site.

IVI.NET drivers cannot be installed until the user has successfully installed the IVI.NET shared components. IVI.NET driver installers have the ability to detect the presence of the IVI.NET shared components and to verify that they are of a version sufficient for the driver. If the IVI.NET shared components are not detected or are not of a sufficient version, the IVI.NET driver installer may call into the IVI.NET shared component installer program, thus providing a seamless install experience for the end user. Alternatively, the IVI.NET driver installer may require the user to run the IVI.NET shared component installer as a separate step before installing the driver. In that case, either the driver supplier distributes the IVI.NET shared component installer with the IVI.NET driver installer, or the IVI.NET driver installer directs the user to the IVI Foundation web site.

To remove the IVI.NET shared components from a system, the user uses the standard Windows Control Panel facility for adding and removing programs.

2.4.2.1 .NET Legacy Installers

Legacy Installers from 2008-2020

Installers from 2008-2020 include multiple .exe packages, one for 32-bit operating systems, and another for 64-bit operating systems. This approach was used for the following versions:

- IVI.NET Shared Components: versions 1.0.0 – 1.4.1.

See section 6.4, *IVI Shared Component Installer Files*, for details.

Note: The IVI Foundation wants to ensure that, if the 32-bit and 64-bit versions of the .NET shared components are on the same machine, they share the same patch level. Allowing the 32-bit IVI.NET shared component installer to run on a 64-bit system might cause a 32-bit version of the .NET shared components to be updated without updating the 64-bit version.

32-bit Only Shared Components (2000-2008)

The IVI Foundation did not support .NET prior to 2010, so the .NET installers have always supported both 32-bit and 64-bit shared components.

2.5 IVI Directory Structure

This section describes the directory structures defined by the IVI Foundation.

32-bit Windows operating systems have a single Program Files directory. 64-bit Windows operating systems have two Program Files directories, one for 64-bit applications and one for 32-bit applications. On 32-bit operating systems the IVI standard root directory exists under the single 32-bit Program Files directory. On 64-bit operating systems the IVI standard root directory exists under both the 32-bit Program Files directory and the 64-bit Program Files directory, for 32-bit and 64-bit applications respectively. The following directory structure diagrams show the IVI standard root directory on 32-bit and 64-bit operating systems.

Windows 7 (32-bit) , Windows 8 (32-bit), and Windows 10 (32-bit)

Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11

Notice that the 32-bit IVI Standard Root Directory path on Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11 is different than it is on Windows 7 (32-bit), Windows 8 (32-bit), and

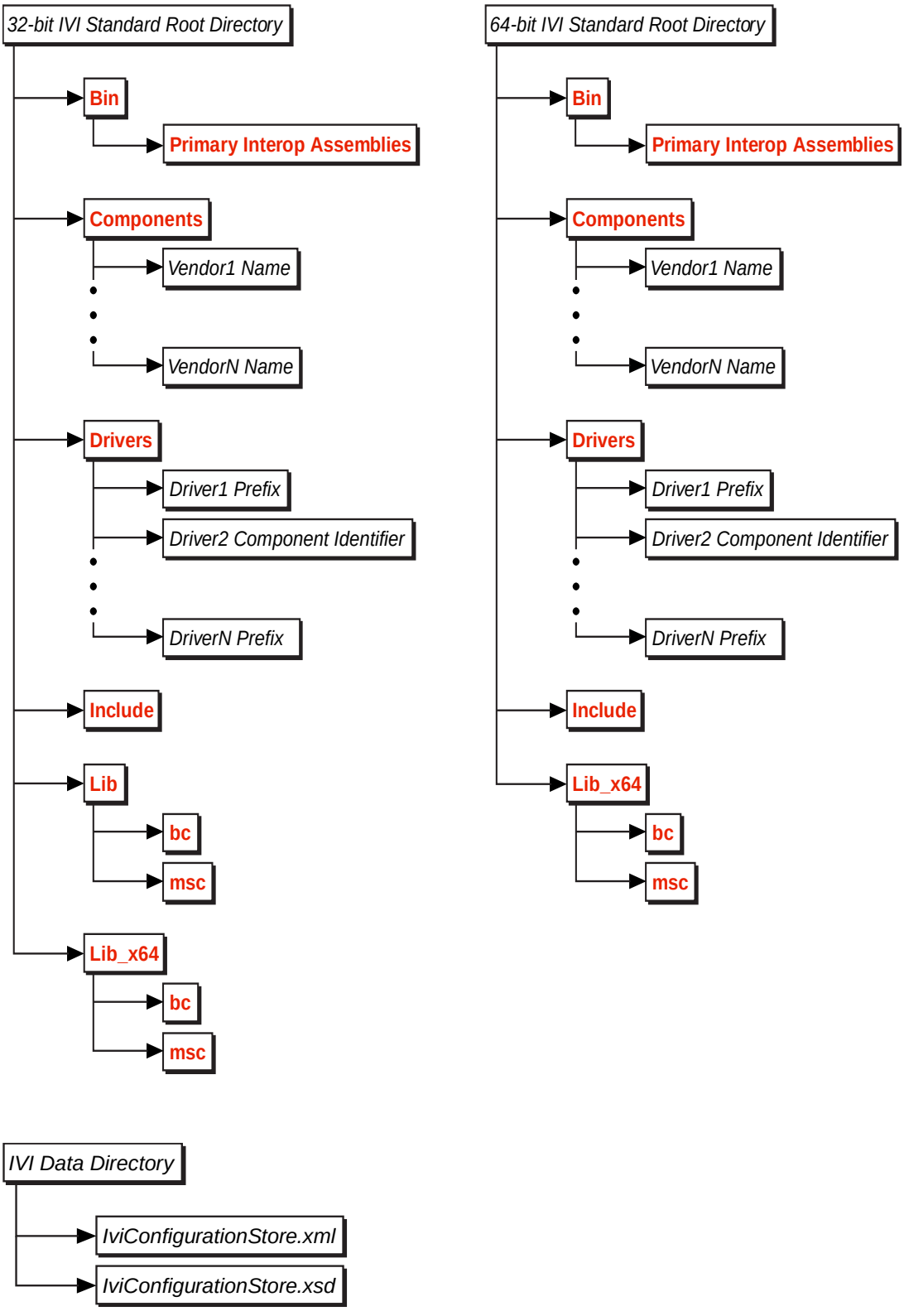
Windows 10 (32-bit). This is because on the 32-bit operating systems the 32-bit Windows Program Files directory name is “Program Files”, whereas on 64-bit Windows it is “Program Files (x86)”.

2.5.1 IVI-COM/IVI-C Directory Structure

Refer to *Appendix A: Example: IVI-COM/IVI-C Driver Installer Scenarios* for an example system on which the user has installed multiple drivers from different vendors.

2.5.1.1 IVI-COM/IVI-C Directory Structure Diagrams

The following diagrams denote the directories that the IVI Foundation specifies. The directories shown in italics are placeholders for names that vendors or users supply.


32-bit IVI Standard Root Directory
Bin
Primary Interop Assemblies
Components
Vendor1 Name
VendorN Name
Drivers
Driver1 Prefix
Driver2 Component Identifier
DriverN Prefix
Include
Lib
bc
msc
Lib_x64
bc
msc
64-bit IVI Standard Root Directory
Bin
Primary Interop Assemblies
Components
Vendor1 Name
VendorN Name
Drivers
Driver1 Prefix
Driver2 Component Identifier
DriverN Prefix
Include
Lib_x64
bc
msc
IVI Data Directory
IviConfigurationStore.xml
IviConfigurationStore.xsd

2.5.1.2 IVI-COM/IVI-C Standard Directory Tree

This section describes the IVI-COM/IVI-C standard directory tree. Driver installations place all files in the IVI standard directory tree, except for files that must be in directories specific to ADEs. The IVI-COM/IVI-C shared components are also installed in the IVI standard directory tree. The root directory of the tree is called the IVI standard root directory.

2.5.1.3 Creation of the IVI-COM/IVI-C Standard Directory Tree

The IVI-COM/IVI-C shared component installer has the ability to detect whether the IVI-COM/IVI-C standard root directory has already been defined in the registry. If it has not been defined, the installer prompts the user for the directory path, creates the IVI standard root directory, creates a registry entry for IVI standard root directory, and creates the standard subdirectories of the IVI standard root directory. The IVI-COM/IVI-C shared component installer, when called from another installer, has the ability to accept the standard root directory path as a command line parameter.

IVI-COM/IVI-C driver installers have the ability to detect whether the IVI standard root directory has already been defined in the registry. If an IVI-COM/IVI-C driver installer detects that the IVI standard root directory has not yet been defined, what it does depends on whether it calls the IVI-COM/IVI-C shared component installer. If it does not call the IVI-COM/IVI-C shared component installer, the IVI-COM/IVI-C driver installer exits and instructs the user to run the IVI shared component installer. If it does call the IVI-COM/IVI-C shared component installer, the IVI-COM/IVI-C driver installer prompts the user to specify a directory path and then passes the directory path to the IVI-COM/IVI-C shared component installer.

2.5.1.4 Contents of the IVI-COM/IVI-C Standard Directory Tree

The root directory of the IVI standard directory tree is intended to contain only subdirectories. IVI-COM/IVI-C installers do not install files directly into the root directory. This section describes the subdirectories in alphabetic order. Note that the `Bin`, `Include`, `Lib`, and `Lib_x64` subdirectories comprise the standard common files directories.

Bin Subdirectory

The `Bin` subdirectory contains all IVI-COM and IVI-C driver DLLs, all IVI-COM/IVI-C shared component DLLs, and all vendor specific shared component DLLs. The `Bin` subdirectory contains the `Primary Interop Assemblies` subdirectory.

All IVI-COM and IVI-C driver DLLs, except .NET Primary Interop Assemblies (PIAs), are installed in the `Bin` subdirectory. PIAs and their corresponding XML IntelliSense help files are installed in the `Bin\Primary Interop Assemblies` subdirectory. The IVI-COM/IVI-C shared component installer creates the `Bin` and `Bin\Primary Interop Assembly` subdirectories.

Components Subdirectory

The `Components` subdirectory contains the non-dispersed IVI-COM/IVI-C shared component files and the non-dispersed vendor specific shared component files. The IVI-COM/IVI-C shared component files are at the top level of the `Components` directory. The vendor specific component files are in vendor specific subdirectories of the `Components` directory.

The IVI-COM/IVI-C shared component installer creates the `Components` subdirectory.

Refer to Section 5.1.6, *Installation of Vendor Specific Shared Components*, for information on installation of vendor specific shared components.

Drivers Subdirectory

The `Drivers` subdirectory contains the standard driver specific subdirectories. The standard driver specific subdirectories contain the non-dispersed driver files. The name of each standard driver specific subdirectory is the driver prefix or component identifier.

The IVI-COM/IVI-C shared component installer creates the `Drivers` subdirectory.

For IVI-COM and IVI-C drivers, the driver installer creates the driver specific subdirectory for the driver.

Include Subdirectory

The `Include` subdirectory contains all header files, and the GUID definition files (`_i.c`).

The IVI-COM/IVI-C shared component installer creates the `Include` subdirectory.

Lib Subdirectory

The `Lib` subdirectory contains two subdirectories `bc`, and `msc`. The `bc` subdirectory contains all Borland-compatible 32-bit DLL import library files. The `msc` subdirectory contains all Microsoft-compatible 32-bit DLL import library files. The IVI-COM/IVI-C shared component installer creates the `Lib`, `Lib\bc`, and `Lib\msc` subdirectories.

Lib_x64 Subdirectory

The `Lib_x64` subdirectory contains two subdirectories `bc`, and `msc`. The `bc` subdirectory contains all Borland-compatible 64-bit DLL import library files. The `msc` subdirectory contains all Microsoft-compatible 64-bit DLL import library files.

The IVI-COM/IVI-C shared component installer creates the `Lib_x64`, `Lib_x64\bc`, and `Lib_x64\msc` subdirectories.

2.5.1.5 Recommendations for Users

This section contains recommendations for users of IVI-COM/IVI-C installers. Driver suppliers should include these recommendations in help documentation for IVI-COM or IVI-C drivers.

- When installing an IVI-COM or IVI-C driver on top of a different version of the driver, uninstall the previous driver before installing the new driver.
- If you no longer need an IVI-COM or IVI-C driver, run the driver uninstaller.
- If you do not need IVI-COM/IVI-C shared components, use the standard Windows Control Panel facility for adding and removing programs to remove the IVI-COM/IVI-C shared components.
- If you are creating an installer that calls an IVI-COM/IVI-C installer, refer to Section 7, *Installer Interface Requirements*, for details on command line syntax that IVI installers use.

2.5.2 IVI.NET Directory Structure

This section describes the .NET specific directory structures defined by the IVI Foundation.

Refer to *Appendix B: Example: IVI.NET Driver Installer Scenarios* for an example system on which the user has installed multiple drivers from different vendors.

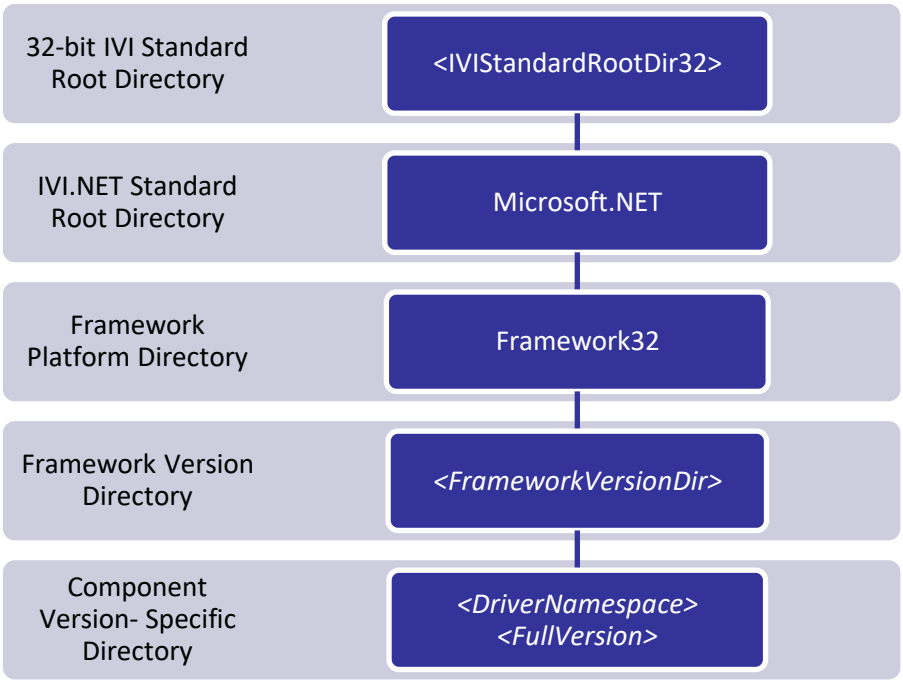
For 32-bit Windows operating systems, the IVI Foundation defines a single IVI.NET standard root directory. For 64-bit Windows operating systems, the IVI Foundation defines two IVI.NET standard root directories, one for 64-bit applications and one for 32-bit applications. On 32-bit operating systems the IVI.NET standard root directory exists under the single 32-bit IVI standard root directory. On 64-bit operating systems the IVI.NET standard root directory exists under both the 32-bit IVI standard root directory and the 64-bit IVI standard root directory, for 32-bit and 64-bit components respectively. For cases where assemblies are compiled as "Any CPU" and are intended to function both as 32-bit and 64-bit components, the assemblies shall be installed in both the 32-bit and 64-bit IVI standard root directories. The following directory structure diagrams show the IVI.NET standard root directory on 32-bit and 64-bit operating systems.

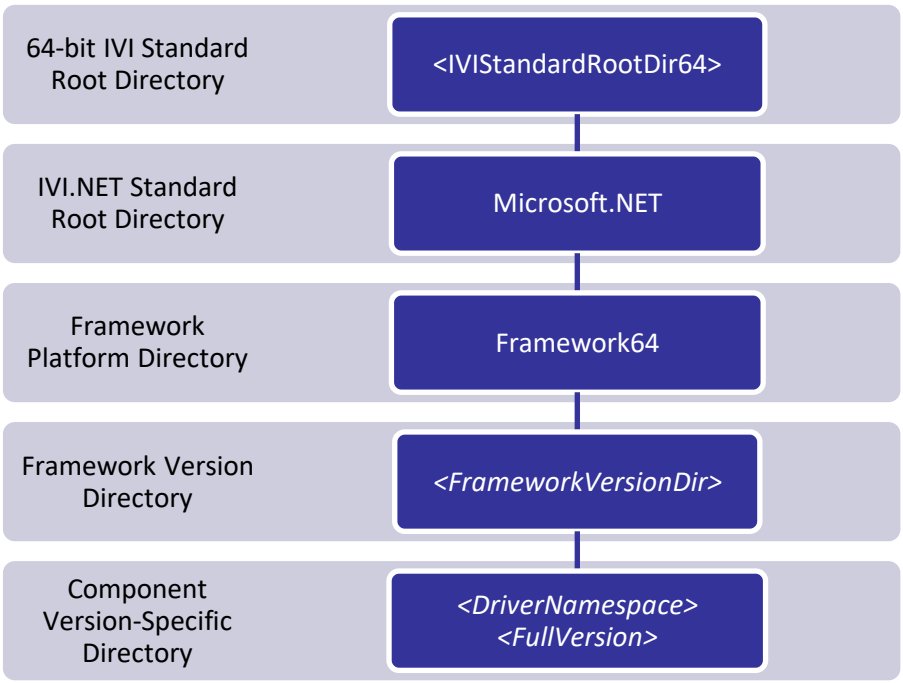
Windows 7 (32-bit) , Windows 8 (32-bit), and Windows 10 (32-bit)

Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11

2.5.2.1 IVI.NET Directory Structure Diagrams

The following diagrams denote the directories that the IVI Foundation specifies. The directories shown in italics are placeholders.





2.5.2.2 IVI.NET Standard Directory Tree

This section describes the IVI.NET standard directory tree. IVI.NET driver and shared component installations place all files in the IVI.NET standard directory tree except for the following files:

- Files that must be in directories specific to ADEs

- Assemblies and policy files that must be in the GAC

The root directory of the tree is called the IVI.NET standard root directory.

2.5.2.3 Creation of the IVI.NET Standard Directory Tree

The IVI.NET shared component installer detects whether the IVI.NET standard root directory has already been defined in the registry. If it has not been defined in the registry, the installer creates the IVI.NET standard root directory, creates a registry entry for IVI.NET standard root directory, and creates the standard directories of the IVI.NET standard root directory.

IVI.NET driver installers detect whether the IVI.NET standard root directory has already been defined in the registry. If an IVI.NET driver installer detects that the IVI.NET standard root directory has not yet been defined, what it does depends on whether it calls the IVI.NET shared component installer. If it does not call the IVI.NET shared component installer, the IVI.NET driver install exits and instructs the user to run the IVI.NET shared component installer. If it does call the IVI.NET shared component installer, the IVI.NET shared component installer creates the IVI.NET standard directory tree and defines the IVI.NET standard root directory in the registry.

2.5.2.4 Contents of the IVI.NET Standard Directory Tree

The root directory of the IVI.NET standard directory tree is intended to contain only subdirectories. IVI.NET installers do not install files directly into the root directory. This section describes the subdirectories in alphabetic order.

Framework32 and Framework64 Directories

The `Framework32` directory tree contains the design-time support that IVI.NET drivers and shared components provide for 32-bit applications. The `Framework64` directory tree contains the design-time support that IVI.NET drivers and shared components provide for 64-bit applications. Components marked as "Any CPU" may be installed to both the `Framework32` and `Framework64` subdirectories to support use in both 32-bit and 64-bit applications.

IVI.NET installers do not install files directly into the `Framework32` and `Framework64` directories. Instead, the files are installed into the `<FrameworkVersionDir>` subdirectories.

The IVI.NET shared component installer creates the `Framework32` and `Framework64` subdirectories.

<FrameworkVersionDir> Subdirectories

The `<FrameworkVersionDir>` subdirectories reside under the Framework32 and Framework64 subdirectories. These subdirectories correspond to released versions of the .NET Framework. Installers install to the particular `<FrameworkVersionDir>` subdirectory tree that corresponds to the minimum major.minor version of the .NET Framework that the installed components require. Each directory name exactly matches the corresponding .NET Framework directory name under `%Windows%\Microsoft.NET\Framework[64]`.

The `<FrameworkVersionDir>` subdirectories contain the following subdirectories:

- IVI.NET Shared Component Version-Specific subdirectories
- IVI.NET Driver Version-Specific subdirectories

The IVI.NET shared component installer creates the `<FrameworkVersionDir>` subdirectories. Refer to Section 4.2.1, *IVI.NET Shared Component Installer Responsibilities*, for additional information on the `<FrameworkVersionDir>` subdirectories that the shared component installer creates.

IVI.NET Component Version-Specific Subdirectories

The IVI.NET component version-specific subdirectories contain the files for the design-time support that IVI.NET drivers and shared components provide.

Each IVI.NET component version-specific subdirectory also contains a version identification file that installers use to determine the presence, vendor, and patch level of an installed IVI.NET driver.

If an IVI.NET driver installer installs unmanaged components, the installer may install such components in any location. The IVI Foundation recommends the following directories:

- WinSxS
- IVI.NET Component Version-Specific subdirectory

2.5.2.5 Recommendations for Users

This section contains recommendations for users of IVI.NET installers. Driver suppliers should include these recommendations in help documentation for IVI.NET drivers.

- IVI.NET driver version numbers are in Major.Minor.Build format.
- IVI.NET driver versions install side-by-side with other versions that have different Major, Minor, or Build numbers. When installing an IVI.NET driver, it is not necessary for you to uninstall versions that have different Major, Minor, or Build numbers.
- If you no longer need the IVI.NET shared components, use the standard Windows Control Panel facility for adding and removing programs to remove the IVI.NET shared components.
- If you are creating an installer that calls an IVI.NET installer, refer to Section 7, *Installer Interface Requirements*, for details on the command line syntax that IVI.NET installers use.

2.6 Wrapper Packaging in IVI Driver Installers

A C wrapper for an IVI-COM driver may be packaged in the same installer or in a different installer.

A COM wrapper for an IVI-C driver may be packaged in the same installer or in a different installer.

A .NET wrapper for an IVI-COM or IVI-C driver shall be packaged in a separate installer.

A COM or C wrapper for an IVI.NET driver shall be packaged in a separate installer.

Note: In this context, “separate installer” means an installer binary that can be run standalone, even though it might also be aggregated with other installers into a single installation program.

3. Requirements for General Behavior of IVI Installers

3.1 Silent and Dialog Installation Modes

IVI driver installers and the IVI shared component installer shall support both dialog mode installation and silent mode installation, with the default being dialog installation mode.

During dialog mode installations, IVI installers shall provide status information interactively. If an error occurs, an installer running in dialog mode displays an error message to the user and may allow the user to make a subsequent choice in an attempt to correct the source of the problem.

3.2 Handling Failures

An IVI installer shall check for all failure conditions that Sections 4, 5, and 6 specifically identify. In addition to these failure conditions, the IVI installer shall check for and report other common installation errors, such as file transfer or access errors, failure to create directories, failure to create registry entries, and failure to modify the Windows search path.

If the IVI installer is run in silent mode and an failure condition occurs, the installer shall reverse the incomplete installation and exit.

If the installer is run in dialog mode and an failure condition occurs, the installer shall take one of the following actions:

- Reverse the incomplete installation, display an informative message to the user, and exit.
- Prompt the user for help in recovering from the failure condition.

Refer to Section 3.4, *Reversing Incomplete Installations*, for details on how IVI installers reverse incomplete installations.

3.3 Handling User Termination of Installer

If the user cancels out of an IVI installer, the IVI installer reverses the incomplete installation and exits. Refer to Section 3.4, *Reversing Incomplete Installations*, for details on how IVI installers reverse incomplete installations.

3.4 Reversing Incomplete Installations

If an IVI installer aborts or fails during installation, the IVI installer shall remove all traces of the partially installed software, including Windows registry entries, IVI configuration store entries, files, and directories. If the software previously existed on the system, the IVI installer shall restore the user's system to its previous state.

If an IVI driver installer calls the IVI shared component installer and a failure occurs after the IVI shared component installer returns, the IVI driver installer shall not reverse the IVI shared component installation.

If an IVI installer calls installers for vendor-defined or third-party components and a failure occurs after installing the components, the reversal process may attempt to remove the components or restore them to their previous state.

Note: If the IVI installer terminates abnormally, such as from a General Protection Fault or other fatal failure condition, the IVI installer might not be able to reverse the incomplete installation.

3.5 Installer Logging

An IVI installer shall give the user or calling program an option for generating an ASCII log file that describes the actions of the installer that changed the state of the user's machine.

4. IVI Directory Structure Creation and Detection Requirements

This section specifies the IVI installer requirements for creating and detecting elements of the IVI directory structure. The responsibilities of the IVI shared component installer and IVI driver installers are discussed separately.

4.1 IVI Standard Root Directory and IVI Data Directory

This section describes the requirements for creating and detecting elements of the IVI standard root directory tree and the IVI data directory.

4.1.1 IVI-COM/IVI-C Shared Component Installer Responsibilities

This section specifies the IVI-COM/IVI-C shared component installer requirements for discovering, registering and creating the IVI standard root directory. The requirements are separated into two parts:

- Responsibilities required by both 32-bit and 64-bit installers
- Additional responsibilities required by the 64-bit installer

4.1.1.1 32-bit and 64-bit IVI-COM/IVI-C Shared Component Installer Responsibilities

The IVI-COM/IVI-C shared component installers, both 32-bit and 64-bit, shall discover, register, and create the 32-bit IVI standard root directory tree according to the following rules. In these rules, `<HKLM\SW>` varies by Operating System and component bitness. Refer to Section 0,
A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.
Determining System Directories and Registry Keys, for details.

1. The installer checks the Windows registry for a non-empty value of the following registry key in the 32-bit registry hive:

   `<HKLM\SW>\IVI, IviStandardRootDir`

2. If `IviStandardRootDir` is already registered, the installer uses the registered root directory for 32-bit shared component installations. If the installer is invoked from the command line with a non-empty 32-bit IVI standard root directory path, the installer ignores the specified path.

   a. The installer checks the Windows registry for a non-empty value of the following registry key:

      `<HKLM\SW>\IVI, IviDataDir`

   b. If `IviDataDir` is not already registered, the installer registers the IVI data directory in the Windows registry as:

      `<HKLM\SW>\IVI, IviDataDir`

      The installer sets the `IviDataDir` value to be `<IviStandardRootDir32>\Data` and then creates the IVI data directory .

      Note: Step 2-b accounts for upgrades of older installations after the user has performed a partial cleanup. In this case, the 32-bit IVI standard root directory was previously defined but not `<IviDataDir>`.

3. If `IviStandardRootDir` is not already registered, the installer takes the following actions.

   a. If the installer is invoked in dialog mode, the installer suggests the following 32-bit IVI standard directory path to the user:

      `<ProgramFilesDir32>\IVI Foundation\IVI`

      The installer allows the user to change the suggested path to another valid path.

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on <ProgramFilesDir32>.

b. A failure condition exists if the installer is invoked in silent mode with an empty or invalid path for the 32-bit IVI standard root directory.

c. A failure condition exists if the path that the user or calling program specifies for the 32-bit IVI standard root directory is one of the following directories or its subdirectories:

- The 32-bit and 64-bit VXI*plug&play* directories. Refer to VXI*plug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the VXI*plug&play* directories.
- **Windows 7 (64-bit), Windows 8 (64-bit), and Windows 10 (64-bit).** The 64-bit IVI standard root directory.
- **Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit) and Windows 11.** The <ProgramFilesDir> directory.

d. The installer registers the 32-bit IVI standard directory path in the Windows registry as:

<HKLM\SW>\IVI, IviStandardRootDir

e. The installer registers the IVI data directory in the Windows registry as:

<HKLM\SW>\IVI, IviDataDir

The IviDataDir path is <ProgramDataDir>\IVI Foundation\IVI.

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on <ProgramDataDir>.

f. The installer checks the Windows registry for a non-empty value of the following registry key:

<HKLM\SW>\IVI\CONFIGURATIONSERVER, MasterStore

g. If MasterStore is not already registered, the installer registers the master configuration store path in the Windows registry as:

<IviDataDir>\IviConfigurationStore.xml

h. The installer adds <IVIStandardRootDir32>\Bin to the Windows system search path.

i. The installer creates a Windows environment variable named IVIROOTDIR32 and sets the value to be the <IVIStandardRootDir32> path.

j. The installer creates the 32-bit IVI standard root directory and the following standard subdirectories: Bin, Bin\Primary Interop Assemblies, Components, Drivers, Include, Lib, Lib\bc, \Lib\msc, Lib_x64, Lib_x64\bc, and \Lib_x64\msc.

k. The installer creates the IVI data directory.

l. If the installer is successful in creating the directories, the installer proceeds.

If the 32-bit IVI standard root directory is defined but the directory or any of the subdirectories do not exist, the IVI shared component installer shall create the missing directories.

4.1.1.2 Additional 64-bit IVI-COM/IVI-C Shared Component Installer Responsibilities

The 64-bit IVI-COM/IVI-C shared component installer shall discover, register, and create the 64-bit IVI standard root directory tree according to the following rules. In these rules, <HKLM\SW> varies by Operating System and component bitness. Refer to Section 0,
A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.
Determining System Directories and Registry Keys, for details.

1. The installer checks the Windows registry for a non-empty value of the following registry key:

   <HKLM\SW>\IVI, IviStandardRootDir

2. If IviStandardRootDir is already registered, the installer uses the registered root directory value for 64-bit shared component installations. If the installer is invoked from the command line with a non-empty 64-bit IVI standard root directory path, the installer ignores the specified path.

3. If IviStandardRootDir is not already registered, the installer takes the following actions.

   a. If the installer is invoked in dialog mode, the installer suggests the following 64-bit IVI standard directory path to the user:

      <ProgramFilesDir>\IVI Foundation\IVI

      The installer allows the user to change the suggested path to another valid path.

      Refer to Section 0,

      A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

      Determining System Directories and Registry Keys, for information on <ProgramFilesDir>.

   b. A failure condition exists if the installer is invoked in silent mode with an empty or invalid path for the 64-bit IVI standard root directory.

   c. A failure condition exists if the path that the user or calling program specifies for the 64-bit IVI standard root directory is one of the following directories or its subdirectories:

      - The 32-bit and 64-bit VXI*plug&play* directories. Refer to VXI*plug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the VXI*plug&play* directories.
      - The 32-bit IVI standard root directory.
      - The <ProgramFilesDir32> directory.

   d. The installer registers the 64-bit IVI standard directory path in the Windows registry as:

      <HKLM\SW>\IVI, IviStandardRootDir

   e. The installer registers the IVI data directory in the Windows registry as:

      <HKLM\SW>\IVI, IviDataDir

      The installer sets the IviDataDir path to the same value as determined in steps 2-a and 2-b of Section 4.1.1.1, *32-bit and 64-bit IVI-COM/IVI-C Shared Component Installer Responsibilities*.

      Note: On 64-bit operating systems, there are two registry keys for IviDataDir (one for 32-bit applications and one for 64-bit applications) whereas there is just one IVI data directory path. Both registry keys shall be set to the same value so that applications of different bitness can easily retrieve the same IVI data directory path.

   f. The installer checks the Windows registry for a non-empty value of the following registry key:

      <HKLM\SW>\IVI\CONFIGURATIONSERVER, MasterStore

   g. If MasterStore is not already registered, the MasterStore path is set to the value determined in steps 4-e and 4-f of Section 4.1.1.1, *32-bit and 64-bit IVI-COM/IVI-C Shared Component Installer Responsibilities*.

Note: On 64-bit operating systems, there are two registry keys for `MasterStore` (one for 32-bit applications and one for 64-bit applications) whereas there is just one master configuration store path. It is important that both registry keys be set to the same value so that applications of different bitness can easily retrieve the same IVI master configuration store path.

h. The installer adds `<IVIStandardRootDir64>\Bin` to the Windows system search path.

i. The installer creates a Windows environment variable named IVIROOTDIR64 and sets the value to be the `<IVIStandardRootDir64>` path.

j. The installer creates the IVI standard root directory and the following standard subdirectories: `Bin`, `Bin\Primary Interop Assemblies`, `Components`, `Drivers`, and `Include`, `Lib_x64`, `Lib_x64\bc`, and `Lib_x64\msc`.

If the installer is successful in creating the directories, the installer proceeds. If the IVI standard root directory is defined but the directory or any of the subdirectories do not exist, the IVI shared component installer should create the missing directories.

4.1.2 IVI-COM/IVI-C Driver Installer Responsibilities

This section specifies the IVI driver installer requirements for detecting the IVI standard root directory. The requirements differ based on the installer type. The different requirements are contained in the subsections below. The following table indicates the subsection that contains the requirements applicable to each installer type.

Table 4-1. Applicable specification sections based on installer type

Installer Type	Sections that Apply
Singular 32-bit driver installer (32-bit OS)	4.1.2.1, *Driver Installer Responsibilities on 32-bit Operating Systems*
Singular 32-bit driver installer (64-bit OS)	4.1.2.2, *32-bit Driver Installer Responsibilities on 64-bit Operating Systems*
Singular 32-bit driver installer (32-bit OS/64-bit OS)	4.1.2.1, *Driver Installer Responsibilities on 32-bit Operating Systems*4.1.2.2, *32-bit Driver Installer Responsibilities on 64-bit Operating Systems*
Singular 64-bit driver installer (64-bit OS)	4.1.2.3, *64-bit Driver Installer Responsibilities*
Unified 32-bit/64-bit driver installer	4.1.2.2, *32-bit Driver Installer Responsibilities on 64-bit Operating Systems* 4.1.2.3, *64-bit Driver Installer Responsibilities*

4.1.2.1 Driver Installer Responsibilities on 32-bit Operating Systems

A driver installer that installs on a 32-bit operating system shall detect the 32-bit IVI standard root directory by checking for a non-empty value of the following registry key:

`<HKLM\SW>\IVI, IviStandardRootDir`

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on `<HKLM\SW>`.

If the IVI-COM/IVI-C driver installer calls the IVI-COM/IVI-C shared component installer and the `IviStandardRootDir` is not already registered, the IVI-COM/IVI-C driver installer takes the following actions:

1. If the IVI-COM/IVI-C driver installer is invoked in dialog mode, the installer suggests the following 32-bit IVI standard root directory path to the user:

   `<ProgramFilesDir32>\IVI Foundation\IVI`

   The installer allows the user to change the suggested path to another valid path.

   Refer to Section 0,

   A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

   Determining System Directories and Registry Keys, for information on `<ProgramFilesDir32>`.

2. A failure condition exists if the IVI-COM/IVI-C driver installer is invoked in silent mode with an empty or invalid path for the 32-bit IVI standard root directory.

3. A failure condition exists if the path that the user or calling program specifies for the 32-bit IVI standard root directory is one of the following directories or its subdirectories:

   - The 32-bit VXI*plug&play* directory. Refer to VXI*plug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the VXI*plug&play* directory.

   Note: This behavior is required so that the IVI-COM/IVI-C driver installer can pass a valid 32-bit IVI standard root directory path when it invokes the IVI-COM/IVI-C shared component installer in silent mode. The IVI shared component installer cannot return errors, so validating the 32-bit IVI standard root directory path beforehand ensures a better user experience.

4.1.2.2 32-bit Driver Installer Responsibilities on 64-bit Operating Systems

A 32-bit driver installer that installs on a 64-bit operating system shall detect the 32-bit IVI standard root directory by checking for a non-empty value of the following registry key:

`<HKLM/SW>\IVI, IviStandardRootDir`

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on `<HKLM/SW>`.

If the IVI-COM/IVI-C driver installer calls the IVI-COM/IVI-C shared component installer and the `IviStandardRootDir` is not already registered, the IVI-COM/IVI-C driver installer takes the following actions:

1. If the IVI-COM/IVI-C driver installer is invoked in dialog mode, the installers suggests the following 32-bit IVI standard root directory path to the user:

   `<ProgramFilesDir32>\IVI Foundation\IVI`

   The installer allows the user to change the suggested path to another valid path.

   Refer to Section 0,

   A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

   Determining System Directories and Registry Keys, for information on `<ProgramFilesDir32>`.

2. A failure condition exists if the IVI-COM/IVI-C driver installer is invoked in silent mode with an empty or invalid path for the 32-bit IVI standard root directory.

3. A failure condition exists if the path that the user or calling program specifies for the 32-bit IVI standard root directory is one of the following directories or its subdirectories:

   - The 32-bit and 64-bit VXI*plug&play* directories. Refer to VXI*plug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the VXI*plug&play* directories.

- The 64-bit IVI standard root directory.
- The <ProgramFilesDir> directory. (Note: This is the 64-bit Program Files directory)

Note: This behavior is required so that the IVI-COM/IVI-C driver installer can pass a valid 32-bit IVI standard root directory path when it invokes the IVI-COM/IVI-C shared component installer in silent mode. The IVI-COM/IVI-C shared component installer cannot return errors, so validating the 32-bit IVI standard root directory path beforehand ensures a better user experience.

4.1.2.3 64-bit Driver Installer Responsibilities

A 64-bit driver installer shall detect the 64-bit IVI standard root directory by checking for a non-empty value of the following registry key:

<HKLM\SW>\IVI, IviStandardRootDir

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on <HKLM\SW>.

If the IVI-COM/IVI-C driver installer calls the IVI-COM/IVI-C shared component installer and the IviStandardRootDir is not already registered, the IVI-COM/IVI-C driver installer takes the following actions:

1. If the IVI-COM/IVI-C driver installer is invoked in dialog mode, the installer suggests the following 64-bit IVI standard root directory path to the user:

   <ProgramFilesDir>\IVI Foundation\IVI

   The installer allows the user to change the suggested path to another valid path.

   Refer to Section 0,

   A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

   Determining System Directories and Registry Keys, for information on <ProgramFilesDir>.
2. A failure condition exists if the IVI-COM/IVI-C driver installer is invoked in silent mode with an empty or invalid path for the 64-bit IVI standard root directory.
3. A failure condition exists if the path that the user or calling program specifies for the 64-bit IVI standard root directory is one of the following directories or its subdirectories:
   - The 32-bit and 64-bit VXI*plug&play* directories. Refer to VXI*plug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the VXI*plug&play* directories.
   - The 32-bit IVI standard root directory.
   - The <ProgramFilesDir32> directory.

   Note: This behavior is required so that the IVI-COM/IVI-C driver installer can pass a valid 64-bit IVI standard root directory path when it invokes the IVI shared component installer in silent mode. The IVI shared component installer cannot return errors, so validating the 64-bit IVI standard root directory path beforehand ensures a better user experience.

4.2 IVI.NET Standard Root Directory

This section describes the requirements for creating and detecting elements of the IVI.NET standard root directory tree.

4.2.1 IVI.NET Shared Component Installer Responsibilities

This section specifies the IVI.NET shared component installer requirements for discovering, registering and creating the IVI.NET standard root directories. The requirements are separated into two parts:

- Responsibilities required by both 32-bit and 64-bit installers
- Additional responsibilities required by the 64-bit installer

4.2.1.1 32-bit and 64-bit IVI.NET Shared Component Installer Responsibilities

The IVI.NET shared component installers, both 32-bit and 64-bit, shall discover, register and create the 32-bit IVI.NET standard root directory tree according to the following rules. In these rules, `<HKLM\SW>` varies by Operating System and component bitness. Refer to Section 0,
A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.
Determining System Directories and Registry Keys, for details.

1. The installer checks the Windows registry for a non-empty value of the following registry key in the 32-bit registry hive:

   `<HKLM\SW>\IVI, IviNetStandardRootDir`

2. If `IviNetStandardRootDir` is already registered, the installer uses the registered root directory for 32-bit IVI.NET shared component installations.

3. If `IviNetStandardRootDir` is not already registered, the installer registers the 32-bit IVI.NET standard root directory in the Windows registry as:

Table 4-2. IVI.NET 32-bit Standard Root Directory Registry Entries

Root	`<HKLM\SW>\IVI`
Key	IviNetStandardRootDir
Value	*Default Value* – `<IviNetStandardRootDir32>`

4. The installer registers design-time assemblies for use with Visual Studio in the Windows registry as described in Section 4.2.3, *Registering IVI.NET Design-Time Assemblies*.

5. The installer creates the 32-bit IVI.NET standard root directory and the following subdirectories:

   a. Framework32

   b. Framework32\`<FrameworkVersionDir>`

      i. The installer creates the `<FrameworkVersionDir>` directory for the minimum major.minor .NET Framework version that the shared components it is installing require. The installer installs the shared components files to this directory.

      ii. The installer creates additional, empty, `<FrameworkVersionDir>` directories for each released version of the .NET Framework that is newer than the minimum major.minor .NET Framework version that the shared components it is installing require. The IVI.NET shared component installer creates these directories so that installers for drivers that depend on newer versions of the .NET Framework do not have to create these directories.

If the installer is successful in creating the directories, the installer proceeds. If the 32-bit IVI.NET standard root directory is defined but the directory or any of the subdirectories do not exist, the IVI.NET shared component installer shall create the missing directories.

4.2.1.2 Additional 64-bit IVI.NET Shared Component Installer Responsibilities

The 64-bit IVI.NET shared component installer shall discover, register, and create the 64-bit IVI.NET standard root directory tree according to the following rules. In these rules, `<HKLM\SW>` varies by Operating System and component bitness. Refer to Section 0,
A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.
Determining System Directories and Registry Keys, for details.

1. The installer checks the Windows registry for a non-empty value of the following registry key:

   `<HKLM\SW>\IVI, IviNetStandardRootDir`

2. If IviNetStandardRootDir is already registered, the installer uses the registered root directory value for 64-bit .NET shared component installations.

3. If `IviNetStandardRootDir` is not already registered, the installer registers the 64-bit IVI.NET standard directory path in the Windows registry as:

Table 4-3. IVI.NET 64-bit Standard Root Directory Registry Entries

Root	`<HKLM\SW>\IVI`
Key	IviNetStandardRootDir
Value	*Default Value* – `<IviNetStandardRootDir64>`

4. The installer creates the 64-bit IVI.NET standard root directory and the following subdirectories:

   a. Framework64

   b. Framework64\`<FrameworkVersionDir>`

      i. The installer creates the `<FrameworkVersionDir>` directory for the minimum major.minor .NET Framework version that the shared components it is installing require. The installer installs the shared components files to this directory.

      ii. The installer creates additional, empty, `<FrameworkVersionDir>` directories for each released version of the .NET Framework that is newer than the minimum major.minor .NET Framework version that the shared components it is installing require. The IVI.NET shared component installer creates these directories so that installers for drivers that depend on newer versions of the .NET Framework do not have to create these directories.

If the installer is successful in creating the directories, the installer proceeds. If the IVI.NET standard root directory is defined but the directory or any of the subdirectories do not exist, the IVI.NET shared component installer should create the missing directories.

4.2.2 IVI.NET Driver Installer Responsibilities

This section specifies the IVI.NET driver installer requirements for detecting the IVI.NET standard root directory. The requirements differ based on the installer type. The different requirements are contained in

the subsections below. The following table indicates the subsection that contains the requirements applicable to each installer type.

Table 4-4. Applicable specification sections based on installer type

Installer Type	Sections that Apply
Singular 32-bit driver installer (32-bit OS)	4.2.2.1, *Driver Installer Responsibilities on 32-bit Operating Systems*
Singular 32-bit driver installer (64-bit OS)	4.2.2.2, *32-bit Driver Installer Responsibilities on 64-bit Operating Systems*
Singular 32-bit driver installer (32-bit OS/64-bit OS)	4.2.2.1, *Driver Installer Responsibilities on 32-bit Operating Systems* 4.2.2.2, *32-bit Driver Installer Responsibilities on 64-bit Operating Systems*
Singular 64-bit driver installer (64-bit OS)	4.2.2.3, *64-bit Driver Installer Responsibilities*
Unified 32-bit/64-bit driver installer	4.2.2.2, *32-bit Driver Installer Responsibilities on 64-bit Operating Systems* 4.2.2.3, *64-bit Driver Installer Responsibilities*

4.2.2.1 Driver Installer Responsibilities on 32-bit Operating Systems

An IVI.NET driver installer that installs on a 32-bit operating system shall detect the 32-bit IVI.NET standard root directory by checking for a non-empty value of the following registry key:

`<HKLM\SW>\IVI, IviNetStandardRootDir`

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on `<HKLM\SW>`.

An IVI.NET driver installer shall check for the presence and version of the IVI.NET shared components as specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*.

An IVI.NET driver installer registers design-time assemblies for use with Visual Studio in the Windows registry as described in Section 4.2.3, *Registering IVI.NET Design-Time Assemblies*.

4.2.2.2 32-bit Driver Installer Responsibilities on 64-bit Operating Systems

A 32-bit IVI.NET driver installer that installs on a 64-bit operating system shall detect the 32-bit IVI.NET standard root directory by checking for a non-empty value for the following registry key:

`<HKLM\SW>\IVI, IviNetStandardRootDir`

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on `<HKLM\SW>`.

An IVI.NET driver installer shall check for the presence and version of the IVI.NET shared components as specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*.

An IVI.NET driver installer registers design-time assemblies for use with Visual Studio in the Windows registry as described in Section 4.2.3, *Registering IVI.NET Design-Time Assemblies*.

4.2.2.3 64-bit Driver Installer Responsibilities

A 64-bit IVI.NET driver installer shall detect the 64-bit IVI.NET standard root directory by checking for a non-empty value for the following registry key:

`<HKLM\SW>\IVI, IviNetStandardRootDir`

Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for information on `<HKLM\SW>`.

An IVI.NET driver installer shall check for the presence and version of the IVI.NET shared components as specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*.

A 64-bit IVI.NET driver installer registers design-time assemblies for use with Visual Studio in the Windows registry as described in Section 4.2.3, *Registering IVI.NET Design-Time Assemblies*.

4.2.3 Registering IVI.NET Design-Time Assemblies

IVI.NET installers shall register design-time assemblies so that the following is the case:

- Microsoft Visual Studio displays the assemblies in the Add References dialog box
- Microsoft Visual Studio and MSBuild can resolve project references to the assemblies.

To achieve this for assemblies compiled as "Any CPU", IVI.NET installers should register design-time assemblies for use with Visual Studio in the 32-bit Windows registry, as follows:

Table 4-2. IVI.NET Design-Time Assemblies Registry Keys

Root	`<HKLM\SW>\Microsoft\.NETFramework\<TargetFrameworkVersion>\AssemblyFoldersEx` Note: this is in the 32-bit Windows registry, regardless of the bitness of the installer or components being registered. This key is defined by Microsoft. Refer to Microsoft documentation for instructions for creating this key, including the specific definition of <TargetFrameworkVersion> in this context.
Key	Component Version-Specific Directory name, followed by the supported Framework version: *For driver installers* – `<DriverNamespace> <FullVersion> (<FwkVerShortName>)` *For the IVI.NET shared component installer* – `IVIFoundationSharedComponents <FullVersion> (<FwkVerShortName>)`
Value	*For 64-bit singular driver installers (64-bit OS)* – Path, under the 64-bit IVI.NET Standard Root Directory, to the Component Version-Specific Directory; or, path to a vendor-specific location with a copy of the assembly identical to the one installed under the IVI.NET Standard Root Directory. *For all other driver installers* – Path, under the 32/-bit IVI.NET Standard Root Directory, to the Component Version-Specific Directory; or, path to a vendor-specific location with a copy of the assembly identical to the one installed under the IVI.NET Standard Root Directory.

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

4.3 Determining System Directories and Registry Keys

This specification uses the term <ProgramFilesDir> to refer to the 32-bit Windows Program Files directory on 32-bit operating systems and the 64-bit Windows Program Files directory on 64-bit operating systems.

The term <ProgramFilesDir32> refers to the 32-bit Windows Program Files directory on all operating systems.

This specification uses the term <ProgramDataDir> to refer to the Windows file system directory containing application data for all users.

The term <HKCR> refers to the location where COM class and type library information is registered. In this specification, the following values should be substituted for <HKCR>, depending on the operating system and the bitness of the COM component.

- 32-bit versions of Windows, 32-bit COM components: <HKCR> = HKEY_CLASSES_ROOT.
- 64-bit versions of Windows, 32-bit COM components: <HKCR> = HKEY_CLASSES_ROOT\Wow6432Node.
- 64-bit versions of Windows, 64-bit COM components: <HKCR> = HKEY_CLASSES_ROOT.

The term <HKLM\SW> refers to the location where HKLM\SOFTWARE information is registered. In this specification, the following values should be substituted for <HKLM\SW>, depending on the operating system and the bitness of the component.

- 32-bit versions of Windows, 32-bit components: <HKLM\SW> = HKEY_LOCAL_MACHINE\SOFTWARE.
- 64-bit versions of Windows, 32-bit components: <HKLM\SW> = HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node.
- 64-bit versions of Windows, 64-bit components: <HKLM\SW> = HKEY_LOCAL_MACHINE\SOFTWARE.

To determine the actual path to these directories and registry key, use the functions provided by your installer tools or the Windows Shell API functions with the identifiers listed in the following table or their equivalents:

Table 4-6. Windows identifiers for operating system directory paths

Directory / Registry Key	Windows 7 (32-bit), Windows 8 (32-bit), and Windows 10 (32-bit)	Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), and Windows 11
<ProgramFilesDir32>	CSIDL_PROGRAM_FILES	CSIDL_PROGRAM_FILESX86
<ProgramFilesDir>	CSIDL_PROGRAM_FILES	CSIDL_PROGRAM_FILES
<ProgramDataDir>	CSIDL_COMMON_APPDATA	CSIDL_COMMON_APPDATA
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node	N/A	KEY_WOW64_32KEY
HKEY_LOCAL_MACHINE\SOFTWARE	N/A	KEY_WOW64_64KEY

Note that Microsoft recommends that applications never access the Wow6432Node key directly as the implementation may change in future releases. Use Windows registry functions that allow use of KEY_WOW64_32KEY and KEY_WOW64_64KEY.

4.4 IVI Shared Component Installer Responsibilities on Windows 7, Windows 8, Windows 10, and Windows 11

On Windows 7, Windows 8, Windows 10, and Windows 11 the IVI shared component installer shall adhere to the following additional rules:

1. If the installer is invoked in dialog mode without admin privileges, the installer shall prompt for elevation. If the installer is invoked in silent mode without admin privileges, a failure condition exists and the installer shall abort.
2. In releases of the IVI-COM/IVI-C shared component installer prior to January 1, 2018, the installer sets the attributes of the IVI standard root directories to disable virtualization and allow modification without admin privileges. In releases after January 1, 2018, the installer shall set the attributes of the IVI standard root directory to require admin privileges for modification. The installer shall do this in the case where the directory already exists and in the case where the installer is creating the directory.

4.5 IVI Driver Installer Responsibilities on Windows 7, Windows 8, Windows 10, and Windows 11

On Windows 7, Windows 8, and Windows 10, if the IVI driver installer calls the IVI shared component installer it shall invoke the IVI shared component installer with admin privileges.

5. IVI Driver Installer Requirements

This section describes the requirements specific to IVI driver installers, other than the requirements described in Section 4, *IVI Directory Structure Creation and Detection Requirements* .

5.1 IVI-COM/IVI-C Driver Installation Procedure

An IVI-COM/IVI-C installer program shall install driver files according to the following procedure:

1. The IVI-COM/IVI-C installer checks the bitness of the Windows operating system and exits with a failure condition if the operating system bitness does not match any of the operating system bitnesses that the installer supports.
2. The IVI-COM/IVI-C installer detects the IVI standard root directory as specified in Section 4.1.1, *IVI-COM/IVI-C Shared Component Installer Responsibilities*.
3. For each supported operating system bitness, if the IVI standard root directory exists, the IVI-COM/IVI-C installer checks for the presence and version of the IVI-COM/IVI-C shared components as specified in Section 5.1.1, *Detecting the Presence and Version of the IVI-COM/IVI-C Shared Components*.
4. For each supported operating system bitness, if the IVI standard root directory does not exist, or the IVI-COM/IVI-C shared components are not installed or not of a sufficient version, the installer takes one of the following two actions:
    a. The IVI-COM/IVI-C installer calls the IVI-COM/IVI-C shared component installer according to the requirements specified in Section 5.1.3, *Calling the IVI-COM/IVI-C Shared Component Installer*. After the IVI-COM/IVI-C shared component installation completes, the IVI-COM/IVI-C installer repeats steps 1 and 2 to verify that the IVI-COM/IVI-C shared component installer completed successfully.
    b. The IVI-COM/IVI-C installer exits with a failure condition. If the installer was invoked in dialog mode, the installer informs the user that the user must first execute the IVI-COM/IVI-C shared component installer and informs the user where to find the IVI-COM/IVI-C shared component installer.
5. The IVI-COM/IVI-C installer checks for the presence, vendor, and version of a previously installed IVI-COM or IVI-C driver of the same name. The installer does this as specified in Section 5.1.2, *Detecting the Presence, Vendor, and Version of an IVI-COM or IVI-C Driver*. If an IVI-COM or driver IVI-C of the same name does exist, the installer takes the following actions. (If drivers of the same name but different bitness exist, the installer repeats these actions for each existing driver.)
    a. A failure condition exists if the vendor of the existing driver does not match the vendor of the driver to be installed or if the vendor of the driver to be installed matches the existing driver but the version of the driver to be installed is less than the version of the existing driver.
    b. If the vendor is the same and the version of the driver to be installed is higher than or equal to the version of the existing driver, the IVI Foundation recommends that the installer proceed with the installation, removing any traces of the previously installed driver while installing the new driver. Alternatively, the installer may exit with a failure condition. If the versions are equal, the driver installer may also exit without a failure condition, run in "repair" mode, or run in "modify" mode.
6. For each supported operating system bitness, the installer creates the standard driver specific directory.
7. For each supported operating system bitness, the installer installs driver files into the appropriate subdirectories of the IVI standard directory tree, as specified in Section 2.5.1.4, *Contents of the IVI-COM/IVI-C Standard Directory Tree*.
8. If any of the driver files are specific to an ADE that requires the files to be in a particular directory outside the IVI standard directory tree, the IVI-COM/IVI-C installer may install such files to that directory. The IVI Foundation recommends that the installation program install such files only if the ADE is present on the system.

9. If the Microsoft .NET Framework exists on the system, then for each supported operating system bitness, the IVI-COM/IVI-C installer shall put the PIAs into the Global Assembly Cache and register each PIA. Refer to Section 3.8, *Legacy PIA Considerations for Drivers*, in *IVI-3.14: Primary Interop Assembly Specification*, for specific files and versions to install.
10. The installer registers the driver with the master IVI configuration store as specified in Section 3.4, *Installing Software Modules*, in *IVI-3.5: IVI Configuration Server Specification*. If a software module entry with the same Name property value as the driver being installed already exists in the IVI configuration store, the installer first deletes the existing software module entry and then re-creates the software module entry. Refer to Section 5.1.4, *IVI-COM/IVI-C Software Module Entries in the IVI Configuration Store*, for how to register software modules.
11. The installer makes any Windows system registry entries that the driver requires as specified in Section 8, *Registry Requirements*.
12. The installer registers an uninstaller program in the standard Windows Control Panel facility for adding and removing programs.
13. If the installer is implemented with MSI technology, the installer shall not set the installed components to be "repaired" automatically.

5.1.1 Detecting the Presence and Version of the IVI-COM/IVI-C Shared Components

An IVI-COM/IVI-C driver installer shall determine the presence of the IVI-COM/IVI-C shared components based on the presence or absence of the `IVISharedComponentVersion.dll` file in the `<IVIStandardRootDir>\Bin` directory.

If the `IVISharedComponentVersion.dll` file exists, the IVI-COM/IVI-C driver installer shall determine the version of the shared components by interrogating the value of FileVersion property of the `IVISharedComponentVersion.dll` file.

5.1.2 Detecting the Presence, Vendor, and Version of an IVI-COM or IVI-C Driver

An IVI-COM/IVI-C driver installer shall determine the presence of an IVI-COM or IVI-C driver based on the presence or absence of the DLL for the driver in the `<IVIStandardRootDir>\Bin` directory. Refer to Section 5.15.10, *Packaging*, in *IVI-3.1: Driver Architecture Specification*, for the DLL filename specifications for IVI-COM drivers. Refer to Section 5.15.10, *Packaging*, in *IVI-3.1: Driver Architecture Specification*, for the DLL filename specifications for IVI-C drivers.

Installers that install on 32-bit operating systems shall check the presence of a 32-bit driver DLL in the `<IVIStandardRootDir32>\Bin` directory. Installers that install on 64-bit operating systems shall check the presence of both a 32-bit driver DLL in the `<IVIStandardRootDir32>\Bin` directory and a 64-bit driver DLL in the `<IVIStandardRootDir64>\Bin` directory.

For each driver DLL file that exists, the installer determines the vendor of the existing driver by interrogating the value of the CompanyName property of the driver DLL file. The installer determines the version of the existing driver by interrogating the value of the FileVersion property of the driver DLL file or by another method that returns the same value as the FileVersion property. Refer to Section 5.18, *File Versioning*, in *IVI-3.1: Driver Architecture Specification*, for details on using the FileVersion property.

Note: The IVI-COM/IVI-C driver installer may utilize a different detection mechanism if the implementation is functionally equivalent to the detection mechanism described in this section.

5.1.3 Calling the IVI-COM/IVI-C Shared Component Installer

An IVI-COM/IVI-C driver installer that calls the IVI-COM/IVI-C shared component installer shall comply with the following rules:

- For each supported operating system bitness, the IVI-COM/IVI-C driver installer detects the IVI standard root directory according to the requirements specified in Section 4.1.2, *IVI-COM/IVI-C Driver Installer Responsibilities*.
- For each supported operating system bitness, if the IVI standard root directory is not already defined, the installer prompts the user to specify a directory path.
- The installer calls the IVI-COM/IVI-C shared component installer with the silent mode command line option. Refer to Section 3.1, *Silent and Dialog Installation Modes*, and Section 7.1, *IVI Shared Component Installer Command Line Syntax*, for more information. For each supported operating system bitness, if the IVI standard root directory is not already defined, the IVI-COM/IVI-C driver installer passes the user-specified directory path to the IVI-COM/IVI-C shared component installer.
- If the IVI-COM/IVI-C shared component installer causes the system to reboot after the IVI-COM/IVI-C shared component installation completes, the IVI-COM/IVI-C driver installer shall resume installation after the system has rebooted.
- The IVI-COM/IVI-C driver installer verifies that the IVI-COM/IVI-C shared component installer completed successfully by taking the following actions:
    - The IVI-COM/IVI-C driver installer detects the IVI standard root directory as specified in Section 4.1.2, *IVI-COM/IVI-C Driver Installer Responsibilities*.
    - If the IVI standard root directory exists, the IVI-COM/IVI-C driver installer checks for the presence and version of the IVI-COM/IVI-C shared components as specified in 5.1.1, *Detecting the Presence and Version of the IVI-COM/IVI-C Shared Components*.
- If the IVI-COM/IVI-C shared components are installed and are of sufficient version, the driver installer proceeds with driver installation.

5.1.4 IVI-COM/IVI-C Software Module Entries in the IVI Configuration Store

An IVI-COM/IVI-C driver installer shall create a software module entry for the driver in the IVI configuration store, as specified in *IVI-3.5: Configuration Server Specification*.

If a C or COM wrapper is installed with the native driver, the driver installer shall not create a separate software module entry for the wrapper.

If a C or COM wrapper is installed separately from the native driver, the wrapper installer shall create a separate software module entry for the wrapper. The name of the software module entry for the wrapper shall be the prefix or component identifier of the native driver followed by "CWrapper" or "COMWrapper".

Note: IVI.NET wrappers shall always be packaged separately from IVI.COM and IVI-C drivers.

Table **5-1** summarizes the requirements for software module attribute values. Software module attribute values shall conform to the values listed in Table **5-1**.

An installer that installs a 32-bit driver shall set the ModulePath32 property to the name of the 32-bit DLL. An installer that installs a 64-bit driver shall set the ModulePath64 property to the name of the 64-bit DLL.

Notice that ModulePath32 and ModulePath64 shall not be the full pathname of the driver DLL for any of the IVI driver package types. Since all IVI driver executables are installed in `<IviStandardRootDir>\Bin`, the full pathname is redundant, and the simple file name of the software module is sufficient.

Table 5-1. Software Module Entries for IVI-COM and IVI-C Drivers

Package Type	Name	ModulePath32/ ModulePath64	Prefix	ProgID	Assembly Qualified Class Name

IVI-C driver	Prefix	File name of software module	Prefix	Empty String	Empty String
IVI-C driver packaged with IVI-COM wrapper	Prefix	File name of software module	Prefix	Version-independent COM ProgID of COM wrapper class	Empty String
IVI-COM driver	Component Identifier	Empty string	Component Identifier	Version-independent COM ProgID of COM wrapper	Empty String
IVI-COM driver packaged with IVI-C wrapper	Prefix of IVI-C wrapper	File name of software module	Prefix of IVI-C wrapper	Version-independent COM ProgID of COM driver	Empty String
IVI-C wrapper packaged separately	Prefix with “CWrapper” appended	File name of software module	Prefix	Empty string	Empty String
IVI-COM wrapper packaged separately	Component Identifier with “COMWrapper” appended	Empty string	Component Identifier	Version-independent COM ProgID of COM wrapper	Empty String

Table **5-2** lists example attribute values for an IVI-COM driver installed with a C wrapper where the IVI-COM component identifier is "Agilent34401A" and the C wrapper's prefix is "Ag34401a".

Table 5-2. Example Software Module Entries for an IVI-COM driver installed with a C wrapper

Software Module Attribute	Value
Name	Ag34401a
ModulePath32	Ag34401a.dll
Prefix	Ag34401a
ProgID	Agilent34401A.Agilent34401A

For additional requirements on creating Software Module Entries in the IVI Configuration Store, refer to Section 5.3, *Details on Software Module Entries in the IVI Configuration Store*.

5.1.5 IVI-COM/IVI-C Driver Uninstaller

The IVI-COM/IVI-C driver installer shall provide a mechanism for the user to uninstall the driver that was previously installed. The uninstaller mechanism shall do the following:

- Remove all Windows registry entries for the driver.
- Removes the PIAs from the Global Assembly Cache and unregisters the PIAs.
- Remove all files for the driver.
- Remove the standard driver specific directory for the driver from the `<IVIStandardRootDir>\Drivers` directory, if it is empty.
- Remove the Software Module entry for the driver from the master IVI configuration store as specified in Section 3.4.2, *Uninstalling Software Modules*, in *IVI-3.5: IVI Configuration Server Specification*.

The uninstaller mechanism shall not do the following:

- Modify or remove IVI-COM/IVI-C shared component files.
- Modify or remove the IVI standard root directory or any of the standard common files directories.
- Modify or remove any IVI-COM/IVI-C driver session configuration entries in the master IVI configuration store.

5.1.6 Installation of Vendor Specific Shared Components

An IVI-COM/IVI-C driver supplier may have components that many of its drivers share. The driver supplier may bundle the installer for these components with each driver or make the installer available separately. If a separate installer is available, it is not an "IVI-COM/IVI-C installer" and is not required to comply with the rules for IVI installers other than the requirements specified in this section.

For any vendor specific shared component files that are installed in the IVI standard root directory tree, the installer shall comply with the following rules:

- The vendor specific shared component files shall be installed *after* the IVI-COM/IVI-C shared components are installed. The installer detects the presence of the IVI-COM/IVI-C shared components according to the requirements specified in 5.1.1, *Detecting the Presence and Version of the IVI-COM/IVI-C Shared Components.* The installer may call the IVI-COM/IVI-C shared component installer as specified in Section 5.1.3, *Calling the IVI-COM/IVI-C Shared Component Installer*.

- DLL files, DLL import library files, and include files shall be installed into the `Bin`, `Lib`, `Lib_x64`, and `Include` subdirectories of the IVI standard root directory. 32-bit DLLs shall be installed in the 32-bit IVI standard root directory tree, and 64-bit DLLs shall only be installed in the 64-bit IVI standard root directory tree. 32-bit import libraries shall only be installed in the 32-bit IVI standard root directory tree. 64-bit import libraries shall be installed in both the 32-bit and 64-bit IVI standard root directory trees. Include files shall be installed in both the 32-bit and 64-bit IVI standard root directory trees.
- For DLLs that drivers or applications find using the Windows search path mechanism, 64-bit DLL names shall differ from 32-bit DLL names.
- Other types of files shall be installed into a subdirectory of the IVI-COM/IVI-C shared component directory tree. The name of the subdirectory should uniquely identify the vendor. If the subdirectory is a two-character prefix, the prefix shall be reserved in the VXI*plug&play* specification *VPP-9: Instrument Vendor Abbreviations*. If the vendor specific subdirectory has not yet been created, the installer shall create it.

5.1.7 Installation of IVI-COM/IVI-C Driver Start Menu Items for Windows 7, Windows 10, and Windows 11

If an IVI-COM/IVI-C Driver installer creates items in the Start Menu, it shall do so according to the following guidelines:

- Subfolders named "IVI" and "IVI Foundation" directly accessible on the start menu shall be reserved for use by components created and maintained by the IVI Foundation.
- The driver installer shall place all start menu shortcuts under a subfolder indicating the instrument driver vendor. For example, an instrument driver supplied by National Instruments would place shortcuts under a "National Instruments" subfolder in the start menu.
- The driver installer may choose to place all start menu shortcuts under an additional subfolder that holds shortcuts related only to that driver. For example, the Agilent Technologies instrument driver for the 34401 digital multimeter may place shortcuts in an "Agilent Technologies IVI Drivers>> Agilent34401" subfolder.
- The driver installer shall not place any start menu shortcuts under a subfolder named "IVI" or "IVI Foundation" unless that subfolder is placed inside another subfolder that identifies the instrument driver vendor. For example, an Agilent Technologies instrument driver for the 34401 digital multimeter may place shortcuts in an "Agilent Technologies Drivers>> IVI" subfolder.
- The IVI Foundation recommends that driver suppliers use the following guidelines for denoting bitness in Start Menu entries for IVI-COM/IVI-C drivers.
    - On 32-bit operating systems, Start Menu entries do not denote bitness.
    - On 64-bit operating systems, Start Menu entries denote bitness only when it is important for users to distinguish between 32-bit and 64-bit versions of files or folders. Start Menu entries should denote bitness by appending "(32-bit)" to 32-bit files and folder entries and "(64-bit)" to 64-bit files and folder entries.
    - Start Menu entries should not denote bitness when linking to items that are not bit-specific. For example, if the Start Menu has two separate entries for 64-bit and 32-bit help files, but the help files are identical copies from two different directories, then the Start Menu entry should not denote bitness. In this case, an even better solution might be to include one only Start Menu entry for the help file.

Instrument driver installers released prior to January 1, 2007, are not subject to the start menu requirements in this section.

5.1.8 Installation of IVI-COM/IVI-C Driver Start and Apps Screen Tiles for Windows 8, Windows 10, and Windows 11

If an IVI-COM/IVI-C Driver installer creates Windows Start or Apps screen tiles, it shall do so according to the following guidelines:

- Driver installers shall not create Start screen tiles for installed components that are not applications. Driver installers may create Start screen tiles for installed components that are applications.
- Driver installers may optionally create Apps screen tiles for installed components, whether applications or not.
- Apps screen sections named "IVI" and "IVI Foundation" shall be reserved for use by components created and maintained by the IVI Foundation.
- The driver installer shall place all Apps screen tiles under a section indicating the instrument driver vendor. For example, an instrument driver supplied by National Instruments would place tiles under a "National Instruments" section.
- Tile names shall include the driver prefix or component ID.
- On 64-bit operating systems, tile names shall include bitness when it is important for users to distinguish between 32-bit and 64-bit versions of files. Tile names shall denote bitness by appending "(32-bit)" and "(64-bit)".

5.2 IVI.NET Driver Installation Procedure

An IVI.NET driver installer program shall install driver files according to the following procedure:

1. The IVI.NET driver installer checks the bitness of the Windows operating system and exits with a failure condition if the operating system bitness does not match any of the operating system bitnesses that the installer supports.
2. For each supported operating system bitness, the IVI.NET driver installer checks for the presence and version of the IVI.NET shared components as specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*.
3. For each supported operating system bitness, if the IVI.NET shared components of the required version are not installed, the installer takes one of the following two actions:
    a. The IVI.NET driver installer calls the IVI.NET shared component installer according to the requirements specified in Section 5.2.3, *Calling the IVI.NET Shared Component Installer*. After the IVI.NET shared component installation completes, the IVI.NET driver installer repeats steps 1 through 3 to verify that the IVI.NET shared component installer completed successfully.
    b. The IVI.NET driver installer exits with a failure condition. If the installer was invoked in dialog mode, the installer informs the user that the user must first execute the IVI.NET shared component installer and informs the user where to find the IVI.NET shared component installer.
4. The IVI.NET driver installer checks for the presence and vendor of a previously installed IVI.NET driver variant of the same name, .NET Framework Version, and Full Version. The installer does this as specified in Section 5.2.2, *Detecting the Presence and Vendor of an IVI.NET Driver Variant.* If a previously installed IVI.NET driver variant of the same name, .NET Framework Version, and Full Version does exist, the installer takes the following actions. (The driver install shall do this for each bitness it supports..)
    a. A failure condition exists if the vendor of the existing driver variant does not match the vendor of the driver variant to be installed.
5. For each supported operating system bitness, the installer creates the Component Version-Specific Directory for the driver, if it does not already exist. The pathname of the Component Version-Specific Directory for the driver variant is based on the bitness, .NET Framework Version, and Full Version of the variant. Refer to Section 1.3.2, *Definition of IVI.NET Installation Terms*, for the format of the Component Version-Specific Directory.

6. For each supported operating system bitness, the installer installs driver files into the appropriate subdirectories of the IVI.NET standard directory tree, as specified in Section 2.5.2.4, *Contents of the IVI.NET Standard Directory Tree*.
7. The installer shall install any required publisher policy files to the GAC. The publisher policy files cause existing applications to use the version being installed instead of the version with which the applications were built. Driver developers may choose not to ship a policy file if they do not want to upgrade existing client applications to the new driver version automatically.
8. If any of the driver files are specific to an ADE that requires the files to be in a particular directory outside the IVI.NET standard directory tree, the IVI.NET driver installer may install such files to that directory. The IVI Foundation recommends that the installation program installs such files only if the ADE is present on the system.
9. The installer registers the driver with the master IVI configuration store as specified in Section 3.4, *Installing Software Modules*, in *IVI-3.5: IVI Configuration Server Specification*. If a software module entry with the same Name property value as the driver being installed already exists in the IVI configuration store, the installer first deletes the existing software module entry and then re-creates the software module entry. Refer to Section 5.2.4, *IVI.NET Software Module Entries in the IVI Configuration Store*, for instructions on how to register software modules.
10. The installer creates any Windows system registry entries that the driver requires as specified in Section 8, *Registry Requirements*.
11. The installer registers an uninstaller program in the standard Windows Control Panel facility for adding and removing programs.
12. If the installer is implemented with MSI technology, the installer shall not set the installed components to be "repaired" automatically.

5.2.1 Detecting the Presence of an IVI.NET Shared Components Variant

An IVI.NET driver installer shall determine the presence of a specific variant of the IVI.NET shared components based on the presence or absence of the `IviFoundationSharedComponentsVersion.dll` file in the Component Version-Specific Directory for the variant.

If the `IviFoundationSharedComponentsVersion.dll` file exists, the IVI.NET driver installer shall determine the Full Version of the shared components by interrogating the value of FileVersion property of the `IviFoundationSharedComponentsVersion.dll` file.

5.2.2 Detecting the Presence and Vendor of an IVI.NET Driver Variant

An IVI.NET driver installer shall determine the presence of the IVI.NET driver variant that it installs based on the presence or absence of the version identification file for the driver in the Component Version-Specific Directory for the variant. Refer to Section 1.3.2, *Definition of IVI.NET Installation Terms*, for the format of the version identification file name.

If the version identification file exists, the installer shall determine the vendor of the existing driver by interrogating the value of the CompanyName property of the version identification file.

If the version identification file exists, the installer shall determine the Full Version of the existing driver by interrogating the value of the FileVersion property of the version identification file or by another method that returns the same value as the FileVersion property. Refer to Section 5.18, *File Versioning*, in *IVI-3.1: Driver Architecture Specification*, for details on using the FileVersion property.

5.2.3 Calling the IVI.NET Shared Component Installer

An IVI.NET driver installer that calls the IVI.NET shared component installer shall comply with the following rules:

- For each operating system bitness that the IVI.NET driver supports, the IVI.NET driver installer detects the presence the IVI.NET shared components variant that the driver requires, according to the requirements specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*.
- The IVI.NET driver installer calls the IVI.NET shared component installer with the silent mode command line option. Refer to Section 3.1, *Silent and Dialog Installation Modes*, and Section 7.1.2, *IVI.NET Shared Component Installer Command Line Syntax*, for more information.
- If the IVI.NET shared component installer causes the system to reboot after the IVI.NET shared component installation completes, the IVI.NET driver installer shall resume installation after the system has rebooted.
- The IVI.NET driver installer verifies that the IVI.NET shared component installer completed successfully by taking the following actions:
    - The IVI.NET driver installer checks for the presence and version of the required IVI.NET shared components variant as specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*.
- If the required variant of the IVI.NET shared components is installed, the driver installer proceeds with driver installation.

5.2.4 IVI.NET Software Module Entries in the IVI Configuration Store

An IVI.NET driver installer shall create a software module entry for the driver variant in the IVI configuration store, as specified in *IVI-3.5: Configuration Server Specification*. Each driver variant has a separate software module entry.

Note: IVI.NET wrappers shall always be packaged separately from IVI-COM and IVI-C drivers.

Table 5-3. Software Module Entries for IVI.NET Drivers

Package Type	Name	ModulePath32/ModulePath64	Prefix	ProgID	Assembly Qualified Class Name
IVI.NET driver	`<DriverNamespace>` `<FullVersion>` `<FwkVerShortName>`	Empty String	Component Identifier	Empty string	Driver's Assembly Qualified Class Name
IVI.NET wrapper	`<DriverNamespace>"Wrapper"` `<FullVersion>` `<FwkVerShortName>`	Empty String	Component Identifier	Empty string	Driver's Assembly Qualified Class Name

As an example, for an IVI.NET driver variant with an IVI.NET driver namespace of "Agilent.Agilent34401a", a Version of "2.5.0", and a .NET Framework Version of "v3.0" the Software Module Entry name is the following:

```
Agilent.Agilent34401a v2.5.0 Fx30
```

If the driver variant is a wrapper, the name is the following:

```
Agilent.Agilent34401aWrapper v2.5.0 Fx30
```

For additional requirements on creating Software Module Entries in the IVI Configuration Store, refer to Section 5.3, *Details on Software Module Entries in the IVI Configuration Store*.

5.2.5 IVI.NET Driver Uninstaller

The IVI.NET driver installer shall provide a mechanism for the user to uninstall the driver variant that was previously installed. The uninstaller mechanism shall do the following:

- Remove all Windows registry entries for the driver.
- Remove the assemblies and policy files from the GAC.
- Remove all files for the driver variant.
- Remove the Component Version-Specific Directory for the driver variant.
- Remove the Software Module entry for the driver variant from the master IVI configuration store as specified in Section 3.4.2, *Uninstalling Software Modules*, in *IVI-3.5: IVI Configuration Server Specification*.

The uninstaller mechanism shall not do the following:

- Modify or remove IVI.NET shared component files.
- Modify or remove the IVI.NET standard root directory or any of its subdirectories that are not specific to a particular driver..
- Modify or remove any IVI.NET driver session configuration entries in the master IVI configuration store.

5.2.6 Installation of Vendor Specific Shared Components

An IVI.NET driver supplier may have components that many of its drivers share. The driver supplier may bundle the installer for these components with each driver or make the installer available separately. If a separate installer is available, it is not an "IVI installer" and is not required to comply with the rules for IVI installers other than the requirements specified in this section.

For any vendor specific shared component files that are installed in the IVI.NET standard root directory tree, the installer shall comply with the following rules:

- The vendor specific shared component files shall be installed *after* the required version of the IVI.NET shared components are installed. The installer detects the presence of the IVI.NET shared components variant it requires according to the requirements specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*. The installer may call the IVI.NET shared component installer as specified in Section 5.2.3, *Calling the IVI.NET Shared Component Installer*.
- Assemblies shall be installed into the Component Version-Specific Directory and the GAC.

5.2.7 Installation of IVI.NET Driver Start Menu Items on Windows 7, Windows 10, and Windows 11

If an IVI.NET Driver installer creates items in the Start Menu, it shall do so according to the following guidelines:

- IVI.NET driver installers may create Start Menu shortcuts under a driver subfolder that holds shortcuts related only to the driver variant. For example, the Agilent Technologies instrument driver for the 34401 digital multimeter, version 2.5.0, .NET Framework version 3.0 may place shortcuts in an "Agilent Technologies IVI Drivers|Agilent34401|2.5.0|Fx30" subfolder.

- The IVI Foundation recommends that driver suppliers use the following guidelines for denoting bitness in Start Menu entries for IVI.NET drivers.
    - On 32-bit operating systems, Start Menu entries do not denote bitness.
    - On 64-bit operating systems, Start Menu entries denote bitness only when it is important for users to distinguish between 32-bit and 64-bit versions of files or folders. Start Menu entries should denote bitness by appending "(32-bit)" to 32-bit files and folder entries and "(64-bit)" to 64-bit files and folder entries.
    - Start Menu entries should not denote bitness when linking to items that are not bit-specific. For example, if the Start Menu has two separate entries for 64-bit and 32-bit help files, but the help files are identical copies from two different directories, then the Start Menu entry should not denote bitness. In this case, an even better solution might be to include one only Start Menu entry for the help file.

5.2.8 Installation of IVI.NET Driver Start and Apps Screen Tiles for Windows 8, Windows 10, and Windows 11

If an IVI.NET Driver installer creates Windows Start or Apps screen tiles, it shall do so according to the following guidelines:

- Driver installers shall not create Start screen tiles for installed components that are not applications. Driver installers may create Start screen tiles for installed components that are applications.
- Driver installers may optionally create Apps screen tiles for installed components, whether applications or not.
- Apps screen sections named "IVI" and "IVI Foundation" shall be reserved for use by components created and maintained by the IVI Foundation.
- The driver installer shall place all Apps screen tiles under a section indicating the instrument driver vendor. For example, an instrument driver supplied by National Instruments would place tiles under a "National Instruments" section.
- Tile names shall include the driver prefix or component ID and the driver version. The framework version designator (e.g. "Fx30") is optional.
- On 64-bit operating systems, tile names shall include bitness when it is important for users to distinguish between 32-bit and 64-bit versions of files. Tile names shall denote bitness by appending "(32-bit)" and "(64-bit)".

5.3 Details on Software Module Entries in the IVI Configuration Store

5.3.1 Including Published API Collections in the IVI Configuration Store

An driver shall include the following in the Published API collection in the software module entry:

- All drivers: `IviDriver`. The major and minor version of the Published API item shall match the latest version of *IVI-3.2, Inherent Capabilities Specification*, that the driver supports.
- All class-compliant specific drivers: A Published API item for each IVI class specification the driver *implements. The major and minor version of* each Published API item shall match the latest version of the class specification that the driver supports.

For each of the above, the driver shall include a separate Published API item for driver type (IVI-COM, IVI-C, or IVI.NET) it supports.

For example, an IVI-COM driver with the IVI-C wrapper would include the following Published API items:

- `IviDriver, IVI-C, 1,0`
- `IviDmm, IVI-C, 3,0`

- `IviDriver, IVI-COM, 1,0`
- `IviDmm, IVI-COM, 3,0`

An IviDmm class-compliant IVI.NET driver would include the following Published API items:

- `IviDriver, IVI.NET, 1,0`
- `IviDmm, IVI.NET, 3,0`

Refer to Section 9, *IVI Published API Class*, in *IVI-3.5: Configuration Server Specification*, for details on constructing a Published API item.

5.3.2 Including Repeated Capability Identifiers in the IVI Configuration Store

An IVI driver shall include its statically-known physical identifiers in the software module entry. In cases where the driver defines qualified physical identifiers, the qualified name shall be used as the Name property in the IVI Physical Name object.

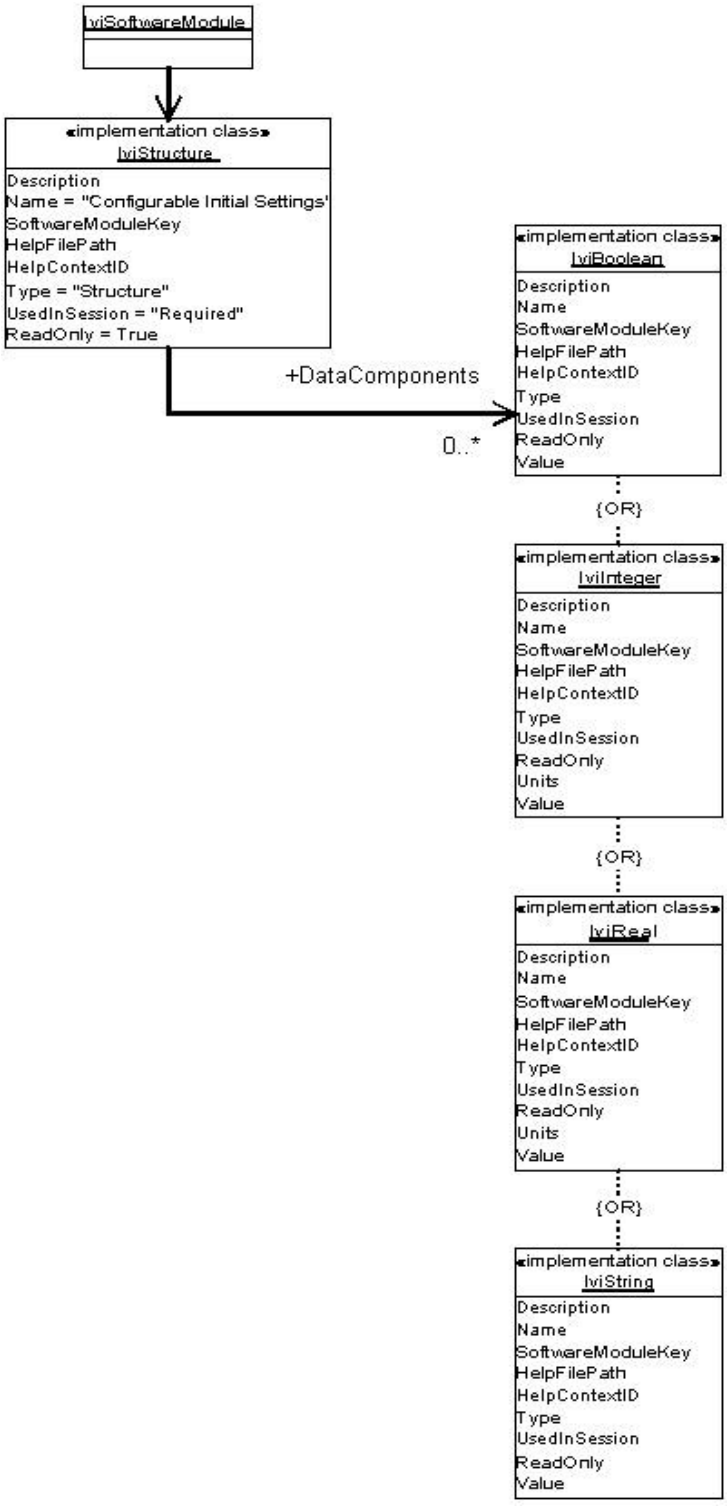
In cases where a repeated capability is defined by an IVI class specification, the RCName in the IVI Physical Name object shall be the repeated capability name as defined in the class specification.

Note: If the driver exports multiple class-compliant interfaces and the class-specifications use different repeated capability names for the same capability, the RCName shall be one of the repeated capability names defined in the class specifications.

5.3.3 Defining Configurable Initial Settings in the IVI Configuration Store

If an IVI driver allows the user to set the initial value of one or more of its attributes through the IVI configuration store, the installation program for the IVI specific driver shall create the IVI configuration store entries as shown in the UML (Unified Modeling Language) object diagram below.

The software module entry for the IVI driver is represented as an IVI Software Module object. The IVI Software Module object contains an IVI Data Components collection. If the driver has one or more configurable initial settings, the installation program for the driver shall create a member of the IVI Data Components collection that is an IVI Structure object with “Configurable Initial Settings” as the value of the Name property. This IVI Structure object shall contain a collection of IVI Data Component objects, each of which represents an attribute setting and has the same data type as the attribute. If the driver has no configurable initial settings the installation program for the driver need not create the “Configurable Initial Settings” object.


IviSoftwareModule
«implementation class»
IviStructure
Description
Name = "Configurable Initial Settings"
SoftwareModuleKey
HelpFilePath
HelpContextID
Type = "Structure"
UsedInSession = "Required"
ReadOnly = True
+DataComponents
0..*
«implementation class»
IviBoolean
Description
Name
SoftwareModuleKey
HelpFilePath
HelpContextID
Type
UsedInSession
ReadOnly
Value
{OR}
«implementation class»
IviInteger
Description
Name
SoftwareModuleKey
HelpFilePath
HelpContextID
Type
UsedInSession
ReadOnly
Units
Value
{OR}
«implementation class»
IviReal
Description
Name
SoftwareModuleKey
HelpFilePath
HelpContextID
Type
UsedInSession
ReadOnly
Units
Value
{OR}
«implementation class»
IviString
Description
Name
SoftwareModuleKey
HelpFilePath
HelpContextID
Type
UsedInSession
ReadOnly
Value

If an attribute applies to a repeated capability, the driver may allow the user to set all instances of the attribute to one value, or the driver may allow the user to set each instance to a different value. If the driver allows the user to set all instances to one value, the installer shall create a single IVI Data Component object for the attribute. If the driver allows the user to set each instance to a different value, the installer shall create a separate IVI Data Component object for each instance.

The following list describes the values to use for the properties in the IVI Data Component objects that represent attribute settings in the IVI Software Module object. Refer to Section 2.8.1, *IVI Data Component*, in *IVI-3.5: IVI Configuration Server Specification*, for more information on the IVI Data Component properties.

- `Description`: A string that describes the intent of the driver developer in making the attribute configurable through the IVI configuration store.
- `Name`: The unique name with which to identify the IVI Data Component object for a particular attribute setting. The name shall be unique within the collection owned by the "Configurable Initial Settings" object. This property shall include the English name of the attribute. If an attribute applies to a repeated capability and the driver allows the user to set each instance of the attribute to a separate value, the name shall also identify the particular instance. For example, the name for the Output Enabled attribute for channel 1 might be "Channel Enabled [Ch1]".
- `SoftwareModuleKey`: A string that the IVI driver uses to identify the attribute and the repeated capability instance, if any. The driver defines the contents of the string. Configuration utility programs should not display the string to the user.
- `HelpFilePath`: The fully specified path to an external help file that contains documentation for the attribute. Usually, this is the help file for the IVI driver. The value of this property may be the empty string if the Description property provides complete information for the attribute.
- `HelpContextID`: The numeric help context ID that points to the section of the help file relevant to the attribute. This property is irrelevant if the Help File Path property is an empty string.
- `Type`: This property is set by the IVI Configuration Server and contains one of the following values depending on the type of the IVI Data Component object: "Boolean", "Real", "Integer", "String", and "APIReference". The "Structure" type shall not be used.
- `Units`: A string that specifies the units for a real or integer attribute. This property is optional and may be an empty string.
- `UsedInSession`: A string indicating whether an IVI Driver Session object that refers to the IVI Software Module object is required to have a copy of IVI Data Component object for the attribute setting. The valid values are "Required" and "Optional". The value "None" shall not be used.

  If the value is "Required", the IVI Configuration Server automatically copies the IVI Data Component object to the IVI Driver Session object that refers to the IVI Software Module object. The configuration utility allows the user to set the value of the attribute to something other than the default value.

  If the value is "Optional", the configuration utility allows the users to decide whether to specify an initial setting for the attribute. If the user decides to specify an initial setting, the configuration utility copies the IVI Data Component object to the IVI Driver Session object, and the configuration utility allows the user to set the value of the attribute to something other than the default value.
- `Value`: A valid default value for the attribute.
- `ReadOnly`: This is always TRUE. End users may not modify anything in a software module entry.

In the copy of the IVI Data Component object that exists in the IVI Driver Session object, the following properties have different meanings or values:

- `Value`: The initial value of the attribute.
- `ReadOnly`: This is always FALSE. The user can modify the Value property through the configuration utility.

After the IVI Data Component objects for the configuration settings are copied to a newly created driver session configuration entry, the configuration utility allows the user to make the following modifications to the driver session configuration entry at any time:

- Change the value of the initial setting of an attribute.
- Remove the IVI Data Component object for an attribute that has “Optional” as the value for the Used In Session property.
- Copy an IVI Data Component object for an attribute from the software module entry if the IVI Data Component object in the software module entry has “Optional” as the value for the Used In Session property.

The configuration utility does not allow the user to make any other modifications to the configurable initial settings in the driver session configuration entry.

6. IVI Shared Component Installer Requirements

This section describes the requirements specific to the IVI shared component installer, other than the requirements described in Section 4, *IVI Directory Structure Creation and Detection Requirements* .

6.1 Overview

This section describes the behavior of the IVI shared component installer that the IVI Foundation provides. The IVI Foundation does not support installation of the IVI shared components other than through the IVI shared component installer that it provides.

6.2 IVI Shared Component Versioning

The 32-bit IVI shared component installer and 64-bit IVI shared component installer shall always be kept at the same version. If a version update to one installer is made, the other installer shall also be rebuilt to have the same version.

6.3 IVI Shared Component Installation

6.3.1 IVI-COM/IVI-C Shared Component Installation

The IVI-COM/IVI-C shared component installer shall install shared component files according to the following procedure:

1. The IVI-COM/IVI-C shared component installer checks the bitness of the Windows operating system. A 32-bit IVI-COM/IVI-C shared component installer exits with a failure condition if the operating system is not a 32-bit operating system. A 64-bit IVI-COM/IVI-C shared component installer exits with a failure condition if the operating system is not a 64-bit operating system.

2. The IVI-COM/IVI-C shared component installer detects and, if necessary, creates the IVI standard root directories according to the requirements specified in Section 4.1.1, *IVI-COM/IVI-C Shared Component Installer Responsibilities*.

3. The IVI-COM/IVI-C shared component installer checks for the presence and version of the IVI-COM/IVI-C shared components as specified in 5.1.1, *Detecting the Presence and Version of the IVI-COM/IVI-C Shared Components*.

4. A failure condition exists if the shared components already exist on the system and have a version greater than the version of the shared components to be installed.

5. If the shared components do not already exist on the system, the installer installs all shared component files to the appropriate directories in the IVI standard directory tree.

    a. Refer to Section 3.1.3, *First Installation*, in *IVI-3.5: Configuration Server Specification* for additional rules on installing the IVI Configuration Server.

    b. The installer registers the IVI-COM shared components on the system.

    c. For each supported operating system bitness, the installer shall install the batch file `IviPiaRegistration.bat` in the `<IVIStandardRootDir>\Bin\Primary Interop Assemblies` directory.

    d. If the Microsoft .NET Framework exists on the system, then for each supported operating system bitness, the IVI shared component installer shall put the PIAs into the Global Assembly Cache and

register each PIA. Refer to Section 2.8, *Legacy PIA Considerations for Shared Components*, in *IVI-3.14: Primary Interop Assembly Specification*, for specific files and versions to install.

6. If the shared components exist on the system, and the version to be installed is greater than that of the shared components already on the system, the installer re-installs all the shared component files except the master configuration store. Refer to Section 3.1.4, *Subsequent Installations*, in *IVI-3.5: Configuration Server Specification* for additional rules on subsequent installations of the IVI configuration server.

7. If the shared components exist on the system, and the version to be installed is equal that of the shared components already on the system, the installer runs in "repair" mode.

8. The IVI-COM/IVI-C shared component installer upgrades a legacy IVI-COM/IVI-C shared component installer if it exists on the system.

9. The IVI-COM/IVI-C shared component installer requires the user to accept the IVI Foundation license to proceed; otherwise the installer will not install the software.

10. The IVI-COM/IVI-C shared component installer installs the IVI-COM/IVI-C Shared Component Cleanup Utility and registers it with the standard Windows Control Panel facility for adding and removing programs.

11. If necessary, the IVI-COM/IVI-C shared component installer reboots the system.

6.3.1.1 IVI-COM/IVI-C Shared Component Cleanup Utility Requirements (V.2.6.x and Prior)

The shared component cleanup utility shall have two modes: *partial cleanup* and *full cleanup*.

In partial cleanup mode, the IVI-COM/IVI-C shared component cleanup utility shall perform the following operations:

- Unregisters the IVI-COM shared components.
- Deletes the shared component files for each shared component.
- Removes the `<IVIStandardRootDir>\Bin` from the Windows system search path.

In full cleanup mode, the IVI-COM/IVI-C shared component cleanup utility shall perform the following operations:

- Unregisters the IVI-COM shared components.
- Removes the PIAs from the Global Assembly Cache and unregisters the PIAs.
- Deletes the registry key and entry for `MasterStore`.
- Deletes the registry key and entry for `IviStandardRootDir`.
- Deletes the registry key and entry for `IviDataDir`.
- Removes the `<IVIStandardRootDir>\Bin` from the Windows system search path.
- Deletes the shared component files for each shared component.
- Deletes the following shared component data files: IviConfigurationStore.xml and IviConfiguration Store.xsd.
- Deletes empty IVI standard common files directories.
- Deletes the IVI-COM/IVI-C shared component directory, if it is empty.
- Deletes the IVI standard root directory, if it is empty.

- Deletes the IVI data directory, if it is empty.
- Unregisters the IVI-COM/IVI-C Shared Component Cleanup Utility with the standard Windows Control Panel facility for adding and removing programs.

Note that the cleanup utility should not be used to remove v.3.0.0 or later. For v.3.0.0 or later, always uninstall the shared components using conventional means.

6.3.2 IVI.NET Shared Component Installation

An IVI.NET shared component installer shall install a shared component variant according to the following procedure:

1. The IVI.NET shared component installer checks the bitness of the Windows operating system. A 32-bit IVI.NET shared component installer exits with a failure condition if the operating system is not a 32-bit operating system. A 64-bit IVI.NET shared component installer exits with a failure condition if the operating system is not a 64-bit operating system.

2. The IVI.NET shared component installer checks for the presence of the required .NET Framework version. A failure condition exists if the required .NET Framework version is not present. If the IVI.NET shared component installer was invoked in dialog mode, the IVI.NET shared component installer instructs the user to run the .NET Framework installer as a separate step before installing the IVI.NET shared components.

3. For each supported bitness, the IVI.NET shared component installer checks for the presence and version of the IVI-COM/IVI-C shared components as specified in Section 5.1.1, *Detecting the Presence and Version of the IVI-COM/IVI-C Shared Components*. If the IVI-COM/IVI-C shared components are not installed or not of a sufficient version, the installer exits with a failure condition. If the installer was invoked in dialog mode, the installer informs the user that the user must first execute the IVI-COM/IVI-C shared component installer and informs the user where to find the IVI-COM/IVI-C shared component installer.

4. The IVI.NET shared component installer checks for the presence of the IVI.NET shared components variant that it installs, as specified in Section 5.2.1, *Detecting the Presence of an IVI.NET Shared Components Variant*. The installer takes the following actions for each bitness it supports:

    a. If the IVI.NET shared components variant does not already exist on the system, the installer installs all IVI.NET shared component files to the appropriate directories in the IVI.NET standard directory tree and the GAC.

    b. If the IVI.NET shared components variant already exists on the system, and the Full Version to be installed is different than the Full Version of the variant on the system, the installer installs all IVI.NET shared component files to the appropriate directories in the IVI.NET standard directory tree and the GAC.

    c. If the IVI.NET shared components variant already exists on the system, and the Full Version to be installed is equal to the Full Version of the variant on the system, the installer shall run in "repair" mode.

5. Publisher policy files shall have assembly version numbers and file version numbers. When publisher policy files are installed to the GAC, they are installed in assembly version-specific directories. To ensure every version of a publisher policy file is installed, each version of a publisher policy file shall have a unique installer Component ID (GUID). The publisher policy files cause existing applications to use the version being installed instead of the version with which the applications were built.

6. The IVI.NET shared component installer requires the user to accept the IVI Foundation license to proceed; otherwise the installer will not install the software.

7. The IVI.NET shared component installer registers with the standard Windows Control Panel facility for adding and removing programs.

8. If necessary, the IVI.NET shared component installer reboots the system.

6.3.2.1 IVI.NET Shared Componet Uninstaller

The IVI. NET shared component uninstaller shall perform the following operations:

- Deletes the IVI.NET shared component files for the shared components variant from the GAC and from the IVI.NET directory tree
- Removes the policy files for the shared components variant from the GAC.
- Removes the Windows registry keys and entries for the design time assemblies for the shared components variant.
- Deletes the IVI.NET standard root directory if no files exist in the directory tree.

6.4 IVI Shared Component Installer Files

This section describes requirements for the IVI shared component installer binary (.exe and.msi) files.

6.4.1 IVI Shared Component Installer File Formats

A Windows installer can be in .msi file format or can be in the format of an .exe file that encapsulates several .msi. Installer problems can occur when a standalone .msi file is run on a system without an encapsulating .exe file. For this reason, the IVI-COM/IVI-C shared component installer .msi files and the IVI.NET shared component installer .msi files shall be made available only to members of the IVI Foundation. The IVI-COM/IVI-C shared component installer .exe file and the IVI.NET shared component installer .exe file shall be made publicly available.

6.4.1.1 IVI Shared Component .exe Installers

The IVI Foundation made significant modernizations to the shared components installers in 2021. For the most part, these changes were either cosmetic or invisible to users. The biggest change, reflected in this section, was to combine the 32-bit and 64-bit installers into a single installer that can run on both 32-bit and 64-bit Windows.

These changes were first implemented in the following versions of the shared components:

- IVI Shared Components: version 3.0.0
- IVI.NET Shared Components: version 2.0.0

2021 Installers and Later

There shall be one 32-bit IVI shared component .exe installer that will run on both 32-bit and 64-bit operating systems. On 32-bit operating systems, the installer shall install only 32-bit shared components. On 64-bit operating systems, the installer shall install both 32-bit and 64-bit shared components.

Installers Prior to 2021

There shall be IVI shared component .exe installers that run only on 32-bit operating systems and are capable of installing only 32-bit shared components. There shall be IVI shared component .exe installers that run only on 64-bit operating systems and are capable of installing both 32-bit and 64-bit shared components.

6.4.1.1.1 IVI-COM/IVI-C Shared Component .exe Installer File Name

V.3.0.0 and Later

The file name of the IVI-COM/IVI-C shared component .exe installer shall be

IviSharedComponents_<FullVersionCompressed>.exe

V.2.6.x and Prior

The file name of the IVI-COM/IVI-C shared component .exe installer that runs only on 32-bit operating systems shall be

IviSharedComponents_<FullVersion>.exe

The file name of the IVI-COM/IVI-C Shared Component .exe installer that runs only on 64-bit operating systems shall be

IviSharedComponents64_<FullVersion>.exe

6.4.1.1.2 IVI.NET Shared Component .exe Installer File Name

V.2.0.0 and Later

The file name of the IVI.NET Shared Component .exe installer shall be

IviNetSharedComponents_<FullVersionCompressed>.exe

V.1.4.x and Prior

The file name of the IVI.NET shared component .exe installer that runs only on 32-bit operating systems shall be

IviNetSharedComponents32_<FwkVerShortName>_<FullVersion>.exe

The file name of the IVI.NET Shared Component .exe installer that runs only on 64-bit operating systems shall be

IviNetSharedComponents64_<FwkVerShortName>_<FullVersion>.exe

6.4.1.2 IVI Shared Component .msi Installers

6.4.1.2.1 IVI-COM/IVI-C Shared Component .msi Installer File Names

V.3.0.0 and Later

The file name of the 32-bit IVI-COM/IVI-C shared component .msi installer shall be

IviSharedComponents_<FullVersionCompressed>.ms
i
The file name of the 64-bit IVI-COM/IVI-C shared component .msi installer shall be

IviSharedComponents_<FullVersionCompressed>.msi

V.2.6.x and Prior

The file name of the 32-bit IVI-COM/IVI-C shared component .msi installer shall be

IviSharedComponents_<FullVersion>.ms
i
The file name of the 64-bit IVI-COM/IVI-C shared component .msi installer shall be

IviSharedComponents64_<FullVersion>.msi

6.4.1.2.2 IVI.NET Shared Component .msi Installer File Names

V.2.0.0 and Later

The file name of the 32-bit IVI-COM/IVI-C shared component .msi installer shall be

IviNetSharedComponents_<FullVersionCompressed>.ms
i
The file name of the 64-bit IVI-COM/IVI-C shared component .msi installer shall be

IviNetSharedComponents_<FullVersionCompressed>.msi

V.1.4.x and Prior

The file name of the 32-bit IVI.NET shared component .msi installer shall be

IviNetSharedComponents32_<FwkVerShortName>_<FullVersion>.msi

The file name of the 64-bit IVI.NET shared component .msi installer shall be

IviNetSharedComponents64_<FwkVerShortName>_<FullVersion>.msi

6.4.2 IVI.NET Shared Component Installer Responsibilities

When a new version of the IVI.NET shared components is created, the IVI.NET working group will determine whether the bindingRedirect tag in all of the IVI.NET assembly .config files needs to change, and if so, will make the appropriate changes.

If the major/minor version numbers are not changed, no new policy files are needed.

If the major/minor version numbers are changed, the build and install need to be modified. The nature of the modifications depends on whether the new revision of the IVI.NET Shared Components is backward compatible with the prior versions of the IVI.NET Shared Componets.

- If the IVI.NET Shared Components are backwards compatible with prior versions, new policy files must be created and installed.
- If the IVI.NET Shared Components are not backwards compatible with prior versions, no policy files shall be created or installed.

The IVI.NET working group will notify the build/install engineer(s) whether policy files are needed and if so, if they are new policy files or modified versions of existing policy files.

- If the IVI.NET working group doesn't notify the build/install engineer, the build/install engineer(s) should ask.
- The build/install engineer(s) should notify the IVI.NET working group of any inconsistencies.

7. Installer Interface Requirements

This section describes the requirements for how IVI installers interact with other IVI installers or calling programs.

7.1 IVI Shared Component Installer Command Line Syntax

7.1.1 IVI-COM/IVI-C Shared Component Installer Command-Line Syntax

If the IVI-COM/IVI-C shared component installer is in the MSI file format, it shall accept a command line argument of the following form and order:

```
msiexec.exe /i <pathtomsi> [IVISTANDARDROOTDIR=<path>]
[IVISTANDARDROOTDIR64=<path>] [/q]
```

The Table 7-1 gives a description each command line argument:

Table 7-1. Command Line Syntax for the IVI-COM/IVI-C Shared Component Installer

Argument	Description
`/i`	Install the MSI file specified by `<pathtomsi>`.
`<pathtomsi>`	The fully-specified path to the `IviSharedComponents.msi` file.
`<path>`	The fully-specified path to use for the IVI standard root directory. The path may contain embedded white space and must contain a terminating backslash. For example, `IVISTANDARDROOTDIR="C:\mydir\ivi\"`
`/q`	Silent mode install (that is, no user interface).

In dialog mode, the IVI-COM/IVI-C shared component installer shall ignore the `<path>` argument.

7.1.2 IVI.NET Shared Component Installer Command Line Syntax

If the IVI.NET shared component installer is in the MSI file format, it shall accept a command line argument of the following form and order:

```
msiexec.exe /i <pathtomsi> [/q]
```

The Table 7-1 gives a description each command line argument:

Table 7-2. Command Line Syntax for the IVI.NET Shared Component Installer

Argument	Description
`/i`	Install the MSI file specified by `<pathtomsi>`.
`<pathtomsi>`	The fully-specified path to the `IviNetSharedComponents.msi` file.
`/q`	Silent mode install (that is, no user interface).

7.2 IVI Driver Installer Command Line Capabilities

IVI driver installers shall allow the user to perform the following actions from the command line:

- Standard Directory Installation
- Uninstallation, if the installer contains uninstallation capability.

IVI driver installers shall allow the user to specify the following on the command line:

- Silent mode installation
- Whether to generate a log file
- The path to the log file

8. Registry Requirements

8.1 IVI-COM Registry Requirements

An IVI-COM driver shall support self-registration.

An IVI-COM driver shall add the following entries to the registry when it self-registers. The strings in angular brackets <> shall be replaced by the appropriate string for the particular IVI-COM driver being registered. Refer to Table 8-5. Registration Entry Substitutions for descriptions and examples of the strings found in angular brackets.

Note that when IVI-COM drivers self-register, the Windows COM registration utility (regsvr32.exe) calls driver routines that provide registration information in a form the utility understands. However, the use that the utility makes of the information differs according to the version and bitness of Windows.

ProgID Entries

If ATL is used to create the IVI-COM driver, the ATL wizard creates code that adds the ProgID related entries shown in Table **8-1**.

<HKCR> varies by Operating System and component bitness. Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for details.

Table 8-1. ProgID Registry Entries

Key	Value
<HKCR>\<ProgID>	
	(Default) REG_SZ <Friendly Name>
<HKCR>\<ProgID>\CLSID	
	(Default) REG_SZ <CLSID>
<HKCR>\<V-I ProgID>	
	(Default) REG_SZ <Friendly Name>
<HKCR>\<V-I ProgID>\CLSID	
	(Default) REG_SZ <CLSID>
<HKCR>\<V-I ProgID>\CurVer	
	(Default) REG_SZ <ProgID>

CLSID Entries

If ATL is used to create the IVI-COM driver, the ATL wizard creates code that adds all of the CLSID related entries with the exception of the CATID entries. The developer shall add the code for the CATID entries manually.

<HKCR> varies by Operating System and component bitness. Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for details.

Table 8-2. CLSID Registry Entries

Key	Value
<HKCR>\CLSID\<CLSID>	
	(Default) REG_SZ <Friendly Name>
<HKCR>\CLSID\<CLSID>\InprocServer32	
	(Default) REG_SZ <Executable File Pathname>
	ThreadingModel REG_SZ Apartment
<HKCR>\CLSID\<CLSID>\ProgID	
	(Default) REG_SZ <ProgID>
<HKCR>\CLSID\<CLSID>\Programmable	
	(Default) REG_SZ (value not set)
<HKCR>\CLSID\<CLSID>\TypeLib	
	(Default) REG_SZ <TypeLib GUID>
<HKCR>\CLSID\<CLSID>\VersionIndependentProgID	
	(Default) REG_SZ <V-I ProgID>
<HKCR>\CLSID\<CLSID>\Implemented Categories	
	(Default) REG_SZ (value not set)
<HKCR>\CLSID\<CLSID>\Implemented Categories\<CATID>	
	(Default) REG_SZ (value not set)

Note: Add as many CATID entries as applies to the driver.

Type Library Entries

If ATL is used to create the IVI-COM driver, the ATL wizard creates code that adds every type library related entry. In addition, the ATL wizard creates a set of four interface entries for each interface defined in the type library.

<HKCR> varies by Operating System and component bitness. Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for details.

Table 8-3. Type Library Registry Entries

Key	Value
<HKCR>\TYPELIB\<TypeLib GUID>	
	(Default) REG_SZ (value not set)
<HKCR>\TYPELIB\<TypeLib GUID>\1.0	
	(Default) REG_SZ <TypeLib HelpString>
<HKCR>\TYPELIB\<TypeLib GUID>\1.0\0	
	(Default) REG_SZ (value not set)
<HKCR>\TYPELIB\<TypeLib GUID>\1.0\0\win32	
	Default) REG_SZ <TypeLib File Pathname>
<HKCR>\TYPELIB\<TypeLib GUID>\1.0\Flags	
	(Default) REG_SZ 0
<HKCR>\TYPELIB\<TypeLib GUID>\1.0\HelpDir	
	(Default) REG_SZ <TypeLib Help File Pathname>
<HKCR>\Interface\<IID>	
	(Default) REG_SZ <Interface Name>
<HKCR>\Interface\<IID>\ProxyStubCLSID	
	(Default) REG_SZ <Universal Marshaller CLSID>
<HKCR>\Interface\<IID>\ProxyStubCLSID32	
	(Default) REG_SZ <Universal Marshaller CLSID>
<HKCR>\Interface\<IID>\TypeLib	
	(Default) REG_SZ <TypeLib GUID>

Note: Add as many CATID entries as applies to the driver.

Category Entries

The IVI-COM class-compliant type library DLLs add the component category for the instrument classes. Individual IVI-COM drivers do not need to add these entries.

<HKCR> varies by Operating System and component bitness. Refer to Section 0,

A vendor may optionally register older versions of design-time assemblies that are installed on the system, in the case where multiple versions of the driver are on the system.

Determining System Directories and Registry Keys, for details.

Table 8-4. Category Registry Entries

Key	Value
<HKCR>\Component Categories\<CATID>	
	(Default) REG_SZ (value not set)
	409 REG_SZ <IVI Instrument Class name>

Table 8-5. Registration Entry Substitutions

Substitute Out	Substitute In	Example
<Program>	Program name. This is not specified by the IVI Foundation. If there is only one driver in a DLL, it is recommended that the value be the same as the Component Identifier property that the IVI driver returns.	YB1234
<CoClass>	CoClass name. This value should be the same as the Component Identifier property that the IVI driver returns.	YB1234
<Version>	Positive integer, starting with 1, and increasing by one each time a new version of the program is released. Note that this is not the same as version or revision defined elsewhere in IVI-3.1 or IVI-3.2	1
<ProgID>	<Program>.<CoClass>.<Version>	YB1234.YB1234.1
<V-I ProgID> (Version-Independent ProgID)	<Program>.<CoClass>	YB1234.YB1234

<Friendly Name>	A name for the CoClass that users will readily understand.	YB1234 IVI-COM driver
<CLSID>	The CLSID for the CoClass.	{7AB5F46D-84EB-44E7-A460-699C622F4979}
<CATID>	The CATID of a category supported by the driver.	{8AB5F468-84EB-44E7-A460-699C682F4979}
<Executable File Pathname>	The path to the executable file that contains the CoClass. This value depends on the bitness of the driver.	C:\Program Files\IVI\Bin\YB1234.dll
<TypeLib GUID>	The GUID assigned to type library associated with the CoClass.	{9AB5F469-84EB-44E7-A460-699C692F4579}
<IVI Instrument Class name>	The COM category name as defined in the class specification or the MSS measurements specification.	IviScope
<TypeLib HelpString>	The help string defined in the type library.	YB1234 1.0 Type Library
<TypeLib File Pathname>	The path to the file that contains the type library. This will typically be the same as the <Executable File Pathname>. This value depends on the bitness of the type library file.	C:\Program Files\IVI\Bin\YB1234.dll
<TypeLib Help File Pathname>	The path to the help file that documents the type library. This value depends on the bitness of the driver.	C:\Program Files\IVI\drivers\YB\YB1234.hlp
<Interface Name>	The name of an interface defined in the type library.	IYb1234Measure
<Universal Marshaller CLSID>	The CLSID of the Universal Marshaller.	

8.2 IVI-C Registry Requirements

There are no registry requirements for IVI-C drivers.

8.3 IVI.NET Registry Requirements

Refer to Section 4.2.3, *Registering IVI.NET Design-Time Assemblies*. There are no other registry requirements for IVI.NET drivers.

9. Example Scenarios and Directories

Refer to *Appendix A: Example: IVI-COM/IVI-C Driver Installer* Scenarios, for example scenarios of typical IVI driver installations. Refer to *Appendix B: Example: IVI.NET Driver Installer Scenarios*, for example scenarios of typical IVI.NET driver installations. Refer to *Appendix C: Example: IVI-COM/IVI-C Installation Directories*, for an example system directory on which a user has installed multiple drivers. Refer to *Appendix D: Example: IVI.NET Installation Directories* for an example system directory on which a user has installed IVI.NET drivers and shared components.

Appendix A: Example: IVI-COM/IVI-C Driver Installer Scenarios

The following examples represent typical use scenarios for IVI driver installations and IVI shared component installations, particularly with regard to detecting the presence of the IVI shared components and previously installed drivers.

1. Dialog mode installation of an Agilent 34401A driver developed by Agilent Technologies, where a version of the driver does not yet exist on the system and the installer does not call the IVI shared component installer.
    a) The Agilent 34401A installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state, and then exits without completing installation.
    b) The Agilent 34401A installer checks to see if there is a conflict with another installed driver of the same name by checking the `<IVIStandardRootDir>\Bin` directory for `ag34401a.dll` and `ag34401a_32.dll`. After determining that there is not a conflict with another installed driver, the installer proceeds.
2. Dialog mode installation of an Agilent 34401A driver developed by Agilent Technologies, where a version of the driver does not exist on the system and the installer calls the IVI shared component installer.
    a) If the IVI standard root directory is not defined, the Agilent 34401A installer prompts the user to specify it. If the IVI standard root directory was not defined, then the installer passes the user-specified IVI standard root directory to the IVI shared component installer.
    b) If the IVI standard root directory is already defined, the Agilent 34401A installer checks to see if the IVI shared components are on the system. If they are present and are of a sufficient version, the Agilent 34401A installation proceeds. If not, the installer calls the IVI shared component installer.
    c) The Agilent 34401A installer checks to see if there is a conflict with another installed driver of the same name by checking the `<IVIStandardRootDir>\Bin` directory for `ag34401a.dll` and `ag34401a_32.dll`. After determining that there is not a conflict with another installed driver, the installer proceeds.
3. Installation of the Tektronix TDS30xx driver developed by National Instruments, following a previous installation of a driver developed by Tektronix for the same instrument family. The Tektronix TDS30xx driver installer developed by National Instruments does not call the IVI shared component installer.
    a) The Tektronix TDS30xx installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the Tektronix TDS30xx installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state and then exits without completing installation.
    b) The Tektronix TDS30xx installer checks to see if an existing driver is on the system by checking the `<IVIStandardRootDir>\Bin` directory for `tktds30xx.dll` and `tktds30xx_32.dll`. After detecting the existence of the driver, the installer checks the vendor and version of the driver DLL.
    c) Since the CompanyName of the driver being installed is "National Instruments" and the CompanyName of the existing driver is "Tektronix", the Tektronix TDS30xx installer notifies the user that a driver for the same instrument already exists on the system and that the user must uninstall it before installing the new driver. The installation program restores the user's system to its previous state and then exits without completing installation.
4. Installation of revision 2. 0 of the Tektronix TDS30xx driver developed by National Instruments, following an installation of revision 1.0 of the same driver. The Tektronix TDS30xx driver installer does not call the IVI shared component installer.
    a) The Tektronix TDS30xx installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the Tektronix TDS30xx

installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state, and then exits without completing installation.

b) The Tektronix TDS30xx installer checks to see if an existing driver is on the system by checking the `<IVIStandardRootDir>\Bin` directory for `tktds30xx.dll` and `tktds30xx_32.dll`. After detecting the existence of the driver, the installer checks the vendor and version of the DLL.

c) Since the CompanyName fields match, the installer proceeds to check the FileVersion fields. Since the driver being installed is a newer version, the installation proceeds.

5. Installation of revision 2.0 of the Tektronix TDS30xx driver developed by National Instruments on a computer with Windows 7 64-bit, following an installation of revision 1.0 of the same driver. Revision 2.0 includes both a 32-bit and a 64-bit IVI driver. Revision 1.0 included 32-bit driver support only.

   a) The Tektronix TDS30xx installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the Tektronix TDS30xx installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state, and then exits without completing installation.

   b) The Tektronix TDS30xx installer checks to see if an existing 32-bit driver is on the system by checking the `<IVIStandardRootDir32>\Bin` directory for `tktds30xx.dll` and `tktds30xx_32.dll`. After detecting the existence of the driver, the installer checks the vendor and version of the DLL.

   c) Since the CompanyName field of the driver being installed matches the CompanyName of the existing 32-bit driver, the installer proceeds to check the FileVersion fields. Since the driver being installed is a newer version, the installation proceeds.

   d) The Tektronix TDS30xx installer also checks to see if an existing 64-bit driver is on the system by checking the `<IVIStandardRootDir64>\Bin` directory for `tktds30xx_64.dll`. After determining that there is not another instance of the driver on the system, the installer proceeds.

6. Installation of revision 2.3 of the Tektronix TDS30xx driver developed by National Instruments on a computer with Windows 7 64-bit, following an installation of revision 2.0 of the same driver. Both revision 2.3 and revision 2.0 include a 32-bit and a 64-bit IVI driver.

   a) The Tektronix TDS30xx installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the Tektronix TDS30xx installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state, and then exits without completing installation.

   b) The Tektronix TDS30xx installer checks to see if an existing 32-bit driver is on the system by checking the `<IVIStandardRootDir32>\Bin` directory for `tktds30xx.dll` and `tktds30xx_32.dll`. After detecting the existence of the driver, the installer checks the vendor and version of the DLL.

   c) Since the CompanyName field of the driver being installed matches the CompanyName of the existing 32-bit driver, the installer proceeds to check the FileVersion fields. Since the driver being installed is a newer version, the installation proceeds.

   d) The Tektronix TDS30xx installer also checks to see if an existing 64-bit driver is on the system by checking the `<IVIStandardRootDir64>\Bin` directory for `tktds30xx_64.dll`. After determining that there is not another instance of the driver on the system, the installer proceeds.

   e) Since the CompanyName field of the driver being installed matches the CompanyName of the existing 64-bit driver, the installer proceeds to check the FileVersion fields. Since the driver being installed is a newer version, the installation proceeds.

7. Installation of revision 2.3 of the Tektronix TDS30xx driver developed by National Instruments on a computer with Windows 7 64-bit, following an installation of another instance of the driver on the same

system from a different vendor. Revision 2.3 includes only a 64-bit IVI driver. The previously installed driver included only a 32-bit IVI driver.

a) The Tektronix TDS30xx installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the Tektronix TDS30xx installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state, and then exits without completing installation.

b) The Tektronix TDS30xx installer checks to see if an existing 32-bit driver is on the system by checking the `<IVIStandardRootDir32>\Bin` directory for `tktds30xx.dll` and `tktds30xx_32.dll`. After detecting the existence of the driver, the installer checks the vendor and version of the DLL.

c) Since the CompanyName of the 64-bit driver being installed is "National Instruments" and the CompanyName of the existing 32-bit driver is a different vendor, the Tektronix TDS30xx installer notifies the user that a driver for the same instrument already exists on the system and that the user must uninstall it before installing the new driver. The installation program restores the user's system to its previous state and then exits without completing installation.

8. Installation of revision 2. 3 of the Tektronix TDS30xx driver developed by National Instruments on a computer with Windows 7 64-bit, following an installation of revision 2.0 of the same driver. Revision 2.3 includes only a 64-bit IVI driver. Revision 2.0 included only a 32-bit IVI driver.

a) The Tektronix TDS30xx installer checks to see if the IVI shared components are on the system with sufficient version. If they are present and of a sufficient version, the Tektronix TDS30xx installation proceeds. If not, the installer notifies the user to run the IVI shared component installer, restores the user's system to its previous state, and then exits without completing installation.

b) The Tektronix TDS30xx installer checks to see if an existing 32-bit driver is on the system by checking the `<IVIStandardRootDir32>\Bin` directory for `tktds30xx.dll` and `tktds30xx_32.dll`. After detecting the existence of the driver, the installer checks the vendor and version of the DLL.

c) Since the CompanyName field of the driver being installed matches the CompanyName of the existing 32-bit driver, the installer proceeds to check the FileVersion fields. Since the driver being installed is a newer version, the installation proceeds. (Note: If the revision number of the 64-bit driver being installed were older than the revision number of the existing 32-bit driver, the installer would report an error and exit.)

d) The Tektronix TDS30xx installer also checks to see if an existing 64-bit driver is on the system by checking the `<IVIStandardRootDir64>\Bin` directory for `tktds30xx_64.dll`. After determining that there is not another instance of the driver on the system, the installer proceeds.

e) The installer uninstalls the 32-bit driver and installs the 64-bit driver. Alternatively, the driver installer may notify the user that a previous version of the driver is installed and must be removed before installing the newer version.

Appendix B: Example: IVI.NET Driver Installer Scenarios

The following examples represent typical use scenarios for IVI.NET driver installations and IVI.NET shared component installations, particularly with regard to detecting the presence of the IVI.NET shared components.

1. Dialog mode installation of an Agilent 34401A driver developed by Agilent Technologies, where the installer does not call the IVI.NET shared component installer.
    a) The Agilent 34401A installer checks to see if the required version of the IVI.NET shared components are on the system. If the version exists, the installation proceeds. If not, the installer notifies the user to run the IVI.NET shared component installer, restores the user's system to its previous state, and then exits without completing installation.
2. Dialog mode installation of an Agilent 34401A driver developed by Agilent Technologies, where the installer calls the IVI.NET shared component installer.
    a) The Agilent 34401A installer checks to see if the required version of the IVI.NET shared components are on the system. If the version exists, the installation proceeds. If not, the installer calls the IVI.NET shared component installer.
    b) The IVI.NET shared component installer checks to see if the required version of the IVI-COM/IVI-C shared components are on the system. If the version exists, the installation proceeds. If not, the installer informs the user that the user must first execute the IVI-COM/IVI-C shared component installer and informs the user where to find the IVI-COM/IVI-C shared component installer.
3. Installation of version 1.0.1 of the Tektronix TDS30xx driver developed by National Instruments following an installation of version 1.0.0 of the same driver.
    a) The Tektronix TDS30xx installer installs version 1.0.1 of the driver.
        a. The installer installs the version 1.0.1 design-time assemblies side-by-side with version 1.0.0 into the Component Version-Specific Directory for version 1.0.1.
        b. The installer installs the version 1.0.1 run-time assemblies into the GAC.
        c. The installer installs policy files to the GAC to redirect references to version 1.0.1 instead of version 1.0.0.
4. Installation of version 2.0.0 of the Tektronix TDS30xx driver developed by National Instruments, following an installation of version 1.0.0 of the same driver.
    a) The Tektronix TDS30xx installer installs version 2.0.0 side-by-side with version 1.0.0.

Appendix C: Example: IVI-COM/IVI-C Installation Directories

The following are example systems on which the user has installed an IVI driver.

Table C-1. Installation of a Fluke 45 IVI-C driver on 32-bit Windows

Directory	Files	Comments
C:\ProgramData\ IVI Foundation\IVI	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
C:\Program Files\ IVI Foundation\IVI		IVI Standard root directory
\Drivers		
\fl45	*Source files (.c, .fp, .sub)*	
	Documentation (compliance doc, help doc)	
	Examples	
\Bin	fl45_32.dll	32-bit DLLs
	IviFloat.dll	
	IviCShared.dll	
	Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)	
\Primary Interop Assemblies	*.NET PIAs and corresponding XML IntelliSense help for IVI-COM shared component dlls (e.g., Ivi.Fgen.Interop.dll & Ivi.Fgen.Interop.xml)*	32-bit PIA's
\Include	fl45.h	
	IviFloat.h	
	IviCShared.h	
	Other shared component include files (e.g., IviConfigServer.h)	
\Lib		
\msc	fl45.lib	Microsoft-compatible 32-bit DLL import libraries
	IviFloat.lib	
	IviCShared.lib	
	Other shared component import library files (e.g., IviConfigServer.lib)	
\Lib_x64		
\msc	fl45.lib	Microsoft-compatible 64-bit DLL import libraries
	IviFloat.lib	

	IviCShared.lib	
	Other shared component import library files (e.g., IviConfigServer.lib)	

Table C-2. Installation of a 32-bit and 64-bit Fluke 45 IVI-C driver on 64-bit Windows

Directory	Files	Comments
C:\ProgramData\ IVI Foundation\IVI	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
C:\Program Files\ IVI Foundation\IVI		64-bit IVI standard root directory
\Drivers		
\fl45	*Source files (.c, .fp, .sub)*	
	Documentation (compliance doc, help doc)	
	Examples	
\Bin	fl45_64.dll	64-bit DLLs
	IviFloat_64.dll	
	IviCShared_64.dll	
	Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)	
\Primary Interop Assemblies	*.NET PIAs and corresponding XML IntelliSense help for IVI-COM shared component dlls (e.g., Ivi.Fgen.Interop.dll & Ivi.Fgen.Interop.xml)*	64-bit PIAs
\Include	fl45.h	
	IviFloat.h	
	IviCShared.h	
	Other shared component include files (e.g., IviConfigServer.h)	
\Lib_x64		
\msc	fl45.lib	Microsoft-compatible 64-bit DLL import libraries
	IviFloat.lib	
	IviCShared.lib	
	Other shared component import library files (e.g., IviConfigServer.lib)	
C:\Program Files (x86)\ IVI Foundation\IVI		32-bit IVI standard root directory
\Drivers		
\fl45	*Source files (.c, .fp, .sub)*	

	Documentation (compliance doc, help doc)	
	Examples	
\Bin	fl45.dll	32-bit DLLs
	IviFloat.dll	
	IviCShared.dll	
	Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)	
\Include	fl45.h	
	IviFloat.h	
	IviCShared.h	
	Other shared component include files (e.g., IviConfigServer.h)	
\Lib		
\msc	fl45.lib	Microsoft-compatible 32-bit DLL import libraries
	IviFloat.lib	
	IviCShared.lib	
	Other shared component import library files (e.g., IviConfigServer.lib)	
\Lib_x64		
\msc	fl45.lib	Microsoft-compatible 64-bit DLL import libraries
	IviFloat.lib	
	IviCShared.lib	
	Other shared component import library files (e.g., IviConfigServer.lib)	

Table C-3. Installation of a 32-bit Rohde & Schwarz RsFs IVI-COM driver on 32-bit Windows

Directory	**Files**	**Comments**
C:\ProgramData\ IVI Foundation\IVI	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
C:\Program Files\ IVI Foundation\IVI		32-bit IVI standard root directory
\Drivers		
\RsFs	*Source files, documentation, examples*	Driver specific files may be organized in subdirectories
	Compliance Document	

	Readme	
	RsFs.chi	
	RsFs.chm	
\Bin	RsFs.dll	32-bit DLLs
	Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)	
\Primary Interop Assemblies	*.NET PIAs and corresponding XML IntelliSense help for IVI-COM shared component dlls (e.g., Ivi.Fgen.Interop.dll & Ivi.Fgen.Interop.xml)*	32-bit PIAs
	Rs.RsFs.Interop.dll	
	Rs.RsFs.Interop.xml	
\Include	*Shared component include files (e.g., IviConfigServer.h)*	
	RsFs.h	
	RsFs_i.c	
\Lib		
\msc	*32-bit shared component import library files (e.g., IviConfigServer.lib)*	IVI-COM drivers do not normally have .lib files.
\bc		
\Lib_x64	*64-bit shared component import library files (e.g., IviConfigServer.lib)*	
\msc		

Table C-4. Installation of a 32-bit and 64-bit AgSAn IVI-COM driver with IVI-C Wrapper on 64-bit Windows

Directory	Files	Comments
C:\ProgramData\ IVI Foundation\IVI	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
C:\Program Files\ IVI Foundation\IVI		64-bit IVI standard root directory
\Drivers		
\AgSAn	*Source files, documentation, examples*	Driver specific files may be organized in subdirectories
	AgilentSAn.chi	
	AgilentSAn.chm	
	Additional MS help files such as AgilentSAn.HxS	Help files required for Visual Studio help integration
	AgSAn.fp	C Wrapper function panel
	AgSAn.sub	C Wrapper .sub file
\Bin	*Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)*	64-bit DLLs
	AgSAn_64.dll	Contains IVI-COM driver and C wrapper.
	AgilentSAnBasic_64.dll	Driver specific support library
	ItlTypeLib_64.dll	Vendor specific support library
\Primary Interop Assemblies	*.NET PIAs and corresponding XML IntelliSense help for IVI-COM shared component dlls (e.g., Ivi.Fgen.Interop.dll & Ivi.Fgen.Interop.xml)*	64-bit PIAs
	Agilent.AgilentSAn.Interop.dll	PIA for primary driver DLL
	Agilent.AgilentSAn.Interop.xml	PIA IntelliSense File
	Agilent.AgilentSAn.Basic.Interop.dll	PIA for support DLL
	Agilent.AgilentSAn.Basic.Interop.xml	PIA IntelliSense File
\Include	*Shared component include files (e.g., IviConfigServer.h)*	
	AgSAn.h	
\Lib_64		64-bit DLL import libraries
\msc	*Shared component import library files (e.g., IviConfigServer.lib)*	Microsoft-compatible
	AgSAn_64.lib	IVI-COM drivers with C wrappers will have .lib files for the C wrapper
C:\Program Files (x86)\ IVI Foundation\IVI		32-bit IVI standard root directory

\Drivers		
\AgSAn	*Source files, documentation, examples*	Driver specific files may be organized in subdirectories
	AgilentSAn.chi	
	AgilentSAn.chm	
	Additional MS help files such as AgilentSAn.HxS	Help files required for Visual Studio help integration
	AgSAn.fp	C Wrapper function panel
	AgSAn.sub	C Wrapper .sub file
\Bin	*Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)*	32-bit DLLs
	AgSAn.dll	Contains IVI-COM driver and C wrapper.
	AgilentSAnBasic.dll	Driver specific support library
	ItlTypeLib.dll	Vendor specific support library
\Primary Interop Assemblies	*.NET PIAs and corresponding XML IntelliSense help for IVI-COM shared component dlls (e.g., Ivi.Fgen.Interop.dll & Ivi.Fgen.Interop.xml)*	32-bit PIAs
	Agilent.AgilentSAn.Interop.dll	PIA for primary driver DLL
	Agilent.AgilentSAn.Interop.xml	PIA IntelliSense File
	Agilent.AgilentSAn.Basic.Interop.dll	PIA for support DLL
	Agilent.AgilentSAn.Basic.Interop.xml	PIA IntelliSense File

Table C-5. Installation of a 32-bit and 64-bit AgSAn IVI-COM driver with IVI-C Wrapper on 64-bit Windows (Continued)

Directory	Files	Comments
C:\Program Files (x86)\ IVI Foundation\IVI		32-bit IVI standard root directory
\Include	*Other shared component include files (e.g., IviConfigServer.h)*	C Wrapper header file
	AgSAn.h	
\Lib		32-bit DLL import libraries
\msc	*Other shared component import library files (e.g., IviConfigServer.lib)*	Microsoft-compatible
	AgSAn.lib	IVI-COM drivers with C wrappers have .lib files for the C wrapper
\Lib_x64		64-bit DLL import libraries
\msc	*Other shared component import library files (e.g., IviConfigServer.lib)*	Microsoft-compatible
	AgSAn_64.lib	IVI-COM drivers with C wrappers have .lib files for the C wrapper

Appendix D: Example: IVI.NET Installation Directories

The following are example systems on which the user has installed an IVI.NET driver.

Table D-1. Installation of a 32-bit AgilentSAn IVI.NET driver on 32-bit Windows

Directory	Files	Comments
C:\ProgramData\ IVI Foundation\IVI	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
C:\Program Files\ IVI Foundation\IVI		32-bit IVI standard root directory
\Drivers		
\Bin		
\Components		
\Microsoft.NET		
\Framework32		
\v2.0.50727		
\Agilent.AgSAn 1.0.0	*Assemblies, documentation, examples*	Driver specific files may be organized in subdirectories
	Agilent.AgilentSAn.chi	Documentation file
	Agilent.AgilentSAn.chm	Documentation file
	Agilent.AgilentSAn.dll	Primary driver assembly (32-bit or Any CPU)
	Agilent.AgilentSAn.xml	IntelliSense file
	Agilent.AgilentSAn.Basic.dll	Support assembly (32-bit or Any CPU)
	Agilent.AgilentSAn.Basic.xml	IntelliSense file
	Agilent.AgSAnVersion.dll	Version identification file
(under …\v2.0.50727) **\IVIFoundationSharedComponents 1.1.0**	*Assemblies, documentation*	
	Ivi.Driver.dll	Assembly (32-bit or Any CPU)
	Ivi.Driver.xml	IntelliSense file
	*Other shared component files (e.g., *.dll, *.xml)*	
	IviFoundationSharedComponentsVersion.dll	Version identification file
\v3.0		
\v3.5		
\Include		
\Lib		
\Lib_64		

Table D-2. Installation of a 32-bit and 64-bit AgilentSAn IVI.NET driver on 64-bit Windows

Directory	Files	Comments
C:\ProgramData\ IVI Foundation\IVI	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
C:\Program Files (x86)\ IVI Foundation\IVI		32-bit IVI standard root directory
\Drivers		
\Bin		
\Components		
\Microsoft.NET		
\Framework32		
\v2.0.50727		
\Agilent.AgSAn 1.0.0	*Assemblies, documentation, examples*	Driver specific files may be organized in subdirectories
	Agilent. AgilentSAn.chi	Documentation file
	Agilent. AgilentSAn.chm	Documentation file
	Agilent.AgilentSAn.dll	Primary driver assembly (32-bit or Any CPU)
	Agilent.AgilentSAn.xml	IntelliSense file
	Agilent.AgilentSAn.Basic.dll	Support assembly (32-bit or Any CPU)
	Agilent.AgilentSAn.Basic.xml	IntelliSense file
	Agilent. AgSAnVersion.dll	Version identification file
(under …\v2.0.50727) **\IVIFoundationSharedComponents 1.1.0**	*Assemblies, documentation*	
	Ivi.Driver.dll	Assembly (32-bit or Any CPU)
	Ivi.Driver.xml	IntelliSense file
	*Other shared component files (e.g., *.dll, *.xml)*	
	IviFoundationSharedComponentsVersion.dll	Version identification file

\v3.0		
\v3.5		
\Include		
\Lib		
\Lib_64		
C:\Program Files\ IVI Foundation\IVI		64-bit IVI standard root directory
\Drivers		
\Bin		
\Components		
\Microsoft.NET		
\Framework64		
\v2.0.50727		
\ Agilent.AgSAn 1.0.0	*Assemblies, documentation, examples*	Driver specific files may be organized in subdirectories
	Agilent. AgilentSAn.chi	Documentation file
	Agilent. AgilentSAn.chm	Documentation file
	Agilent.AgilentSAn.dll	Primary driver assembly (64-bit or Any CPU)
	Agilent.AgilentSAn.xml	IntelliSense file
	Agilent.AgilentSAn.Basic.dll	Support assembly (64-bit or Any CPU)
	Agilent.AgilentSAn.Basic.xml	IntelliSense file
	Agilent. AgSAnVersion.dll	Version identification file
(under …\v2.0.50727) **\IVIFoundationSharedComponents 1.1.0**	*Assemblies, documentation*	
	Ivi.Driver.dll	Assembly (64-bit or Any CPU)
	Ivi.Driver.xml	IntelliSense file
	*Other shared component files (e.g., *.dll, *.xml)*	
	IviFoundationSharedComponentsVersion.dll	Version identification file

\v3.0		
\v3.5		
\Include		
\Lib		
\Lib_64		