



## **IVI-3.17: Installation Requirements Specification**

March 30, 2009 Edition  
Revision 1.0

# 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](http://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](http://www.ivifoundation.org).

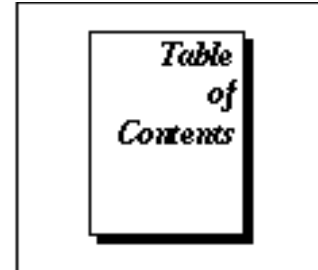
## **Warranty**

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.



---

<b>Important Information .....</b>	<b>2</b>
<b>Warranty            2</b>	
<b>Trademarks        2</b>	
<b>IVI-3.17: Installation Requirements Specification .....</b>	<b>6</b>
<b>1. Overview of the IVI Installation Requirements Specification .....</b>	<b>7</b>
1.1 Introduction.....	7
1.2 Definition of Installation Terms .....	7
<b>2. Features and Intended Use of Installers .....</b>	<b>10</b>
2.1 Introduction.....	10
2.2 Installers.....	10
2.3 IVI Driver Installation .....	10
2.3.1 IVI Drivers Installers and Bitness .....	10
2.3.1.1 IVI Driver Installer Types.....	10
2.3.1.2 Valid Uses of IVI Driver Installer Types .....	11
2.3.1.3 Recommended IVI Driver Installer Approach .....	13
2.4 IVI Shared Component Installation .....	13
2.5 IVI Directory Structure.....	14
2.5.1 IVI Directory Structure Diagrams .....	16
2.5.2 IVI Standard Directory Tree.....	17
2.5.2.1 Creation of the IVI Standard Directory Tree .....	17
2.5.2.2 Contents of the IVI Standard Directory Tree .....	17
2.5.3 Recommendations for Users.....	18
<b>3. Requirements for General Behavior of IVI Installers .....</b>	<b>19</b>
3.1 Silent and Dialog Installation Modes .....	19
3.2 Handling Failures.....	19
3.3 Handling User Termination of Installer.....	19
3.4 Reversing Incomplete Installations.....	19
3.5 Installer Logging.....	20

## **4. Requirements for Creating and Detecting the IVI Directory Structure**

**21**

4.1 IVI Standard Root Directory and IVI Data Directory .....	21
4.1.1 IVI Shared Component Installer Responsibilities .....	21
4.1.1.1 32-bit and 64-bit IVI Shared Component Installer Responsibilities .....	21
4.1.1.2 Additional 64-bit IVI Shared Component Installer Responsibilities .....	22
4.1.2 IVI Driver Installer Responsibilities .....	24
4.1.2.1 Driver Installer Responsibilities on 32-bit Operating Systems .....	24
4.1.2.2 32-bit Driver Installer Responsibilities on 64-bit Operating Systems .....	25
4.1.2.3 64-bit Driver Installer Responsibilities .....	25
4.2 Determining System Directories and Registry Keys .....	26
4.3 IVI Shared Component Installer Responsibilities on Windows Vista.....	27
4.4 IVI Driver Installer Responsibilities on Windows Vista.....	27

## **5. IVI Driver Installer Requirements.....28**

5.1 Driver Installation Procedure.....	28
5.2 Detecting the Presence and Version of the IVI Shared Components .....	29
5.3 Detecting the Presence, Vendor, and Version of an IVI Driver .....	29
5.4 Calling the IVI Shared Component Installer .....	29
5.5 Software Module Entries in the IVI Configuration Store.....	30
5.5.1 Including Published API Collections in the IVI Configuration Store .....	32
5.5.2 Including Repeated Capability Identifiers in the IVI Configuration Store.....	32
5.5.3 Defining Configurable Initial Settings in the IVI Configuration Store .....	33
5.6 Driver Uninstaller .....	36
5.7 Installation of Vendor Specific Shared Components.....	36
5.8 Installation of IVI Driver Start Menu Items .....	37

## **6. IVI Shared Component Installer Requirements .....38**

6.1 Overview.....	38
6.2 IVI Shared Component Versioning .....	38
6.3 IVI Shared Component Installation.....	38
6.4 IVI Shared Component Cleanup Utility Requirements .....	39

## **7. Installer Interface Requirements.....40**

7.1 IVI Shared Component Installer Command Line Syntax .....	40
7.2 IVI Driver Installer Command Line Capabilities .....	40

## **8. Registry Requirements .....41**

8.1 IVI-COM Registry Requirements.....	41
8.2 IVI-C Registry Requirements .....	45

**9. Example Scenarios and Directories .....46**

**Appendix A: Example: IVI Driver Installer Scenarios .....47**

**Appendix B: Example: IVI Installation Directories .....50**

# 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

<b>Revision Number</b>	<b>Date of Revision</b>	<b>Revision Notes</b>
Revision 1.0	March 30, 2009	Original release. Created from the Section 6, <i>Installation Requirements</i> , in <i>IVI-3.1: Driver Architecture Specification</i> .

# 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 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 Shared Component Installer**

An IVI installer created and distributed by the IVI Foundation that installs all the IVI shared component files. When required, the term 32-bit IVI Shared Component Installer will be used to refer to the IVI Shared Component Installer that installs on a 32-bit operating system and the term 64-bit IVI Shared Component Installer will be used to refer to the IVI Shared Component Installer that installs on a 64-bit operating system.

### **IVI Shared Component Cleanup Utility**

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

### **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. 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 2000, Windows XP, and Vista 32 have only a 32-bit IVI standard root directory. Vista 64 has 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 4.2, *Determining System Directories and Registry Keys*, for information on <ProgramFilesDir>.

### **IVI Data Directory <IVIDataDir>**

The IVI data directory contains the master IVI configuration store and other data files installed by the IVI shared component installer. 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 4.2, *Determining System Directories and Registry Keys*, for information on <ProgramDataDir>.

## 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 shared component files as well as IVI driver files.

Note: An installer *dispersed* 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.

## Standard Driver Specific Directory

The directory into which a driver installation 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 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 shared component directory is <IVIStandardRootDir>\Components.

## Dispersed File

An IVI 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 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.

## 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 IVI installer to set installer options. Instead, another installer program calls the IVI installer, passing installer options 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 Vista feature that manages user and application security privileges.

## Standard Privileges

The default UAC privileges on Windows Vista for all applications.

## **Admin Privileges**

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

## 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

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

#### 2.3.1 IVI Drivers Installers and Bitness

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

##### 2.3.1.1 IVI Driver Installer Types

To avoid confusion, this specification uses a naming convention for the various types of IVI driver installers.

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 versions 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 a 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 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.

## 2.3.1.2 Valid Uses of IVI Driver Installer Types

This section describes the set of valid IVI driver installer types based on the operating systems a driver supplier supports and the bitness of IVI drivers the supplier provides.

**Table 2-1.** Valid installer 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	<p><b>Option A:</b> 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.</p> <p><b>Option B:</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 shared component installer. If the installer calls the IVI shared component installer, then this option requires either calling the legacy IVI shared component installer or calling both the 32-bit and 64-bit IVI shared component installers.</p>

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

Supported Operating Systems	Valid Combinations of Driver Installation Program Types
<p>32-bit driver is supported on both 32-bit and 64-bit operating systems</p> <p>64-bit driver is supported on 64-bit operating systems</p>	<p>Singular 32-bit driver installer (32-bit OS)</p> <p>AND</p> <p>Unified 32-bit/64-bit driver installer (64-bit OS)</p> <p>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.</p>
<p>32-bit driver is only supported on 32-bit operating systems</p> <p>64-bit driver is supported on 64-bit operating systems</p> <p>Note: This excludes installation of a 32-bit driver on a 64-bit operating system</p>	<p>Singular 32-bit driver installer (32-bit OS)</p> <p>AND</p> <p>Singular 64-bit driver installer (64-bit OS)</p> <p>Note: The Singular 32-bit driver installer (32-bit OS) shall refuse to install on 64-bit operating systems.</p>
<p>32 bit driver is only supported on 64-bit operating systems</p> <p>64-bit driver is supported on 64-bit operating systems</p>	<p>Unified 32-bit/64-bit driver installer (64-bit OS)</p> <p>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.</p>

### 2.3.1.3 Recommended IVI 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 types:

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

Prior to version 2.0 of this specification, IVI 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.

## 2.4 IVI Shared Component Installation

The IVI Foundation provides three IVI shared components installation programs:

- *64-bit IVI shared component installer*, which installs both 32-bit and 64-bit IVI shared components on 64-bit operating systems. The 64-bit installer refuses to install on 32-bit operating systems.
- *32-bit IVI shared component installer*, which installs 32-bit IVI shared components on 32-bit operating systems. The 32-bit installer refuses to install on 64-bit operating systems.

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

- *Legacy IVI shared component installer*, which installs 32-bit IVI shared components on both 32-bit and 64-bit operating systems. The legacy IVI shared component installer has a version of 1.5.1 or less. The IVI Foundation will distribute version 1.5.1 of the legacy IVI shared component installer, which complied with version 1.7 of this specification, until Microsoft no longer supports Windows XP or until the IVI Foundation determines that no IVI driver installers require the legacy shared component installer. No maintenance updates will be made to the legacy IVI shared component installer.

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 shared component installer. If one of the two first two installers listed above has ever been run successfully, the legacy installer will do nothing.

Each IVI shared component installer program installs the following IVI shared components:

- IVI Configuration Server
- IVI Floating Point Services
- IVI C Shared Components
- IVI COM Session Factory
- IVI Type Libraries
- IVI Primary Interop Assemblies
- IviLxiSync Components

The latest IVI shared component installation programs are also available for public download from the IVI Foundation web site, [www.ivifoundation.org](http://www.ivifoundation.org), and are not distributed as an IVI shared component.

IVI drivers cannot be installed until the user has successfully installed the IVI shared components for the appropriate operating system. IVI 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 driver installer may call into the IVI shared component installer program, thus providing a seamless install experience for the end user. Alternatively, the IVI driver installer may require the user to run the IVI shared component installer as a separate step before installing the driver. In that case, either the driver supplier distributes the IVI shared component installer with the IVI driver installer, or the IVI driver installer directs the user to the IVI Foundation web site.

The IVI 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 shared components from a system, the user either uses the Windows Add Remove Programs facility or runs the IVI shared component cleanup utility that the IVI Foundation provides. The IVI shared component cleanup utility is available for public download from the IVI Foundation web site.

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

## **2.5 IVI Directory Structure**

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

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

On 32-bit Windows operating systems, there is a single Program Files directory. On Windows Vista 64, there are 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 2000, Windows XP, and Windows Vista 32**

C:\Program Files\ IVI Foundation\ IVI\ <i>(IVI Standard Root Directory)</i>	<i>32-bit Directory Structure</i>
---	-----------------------------------

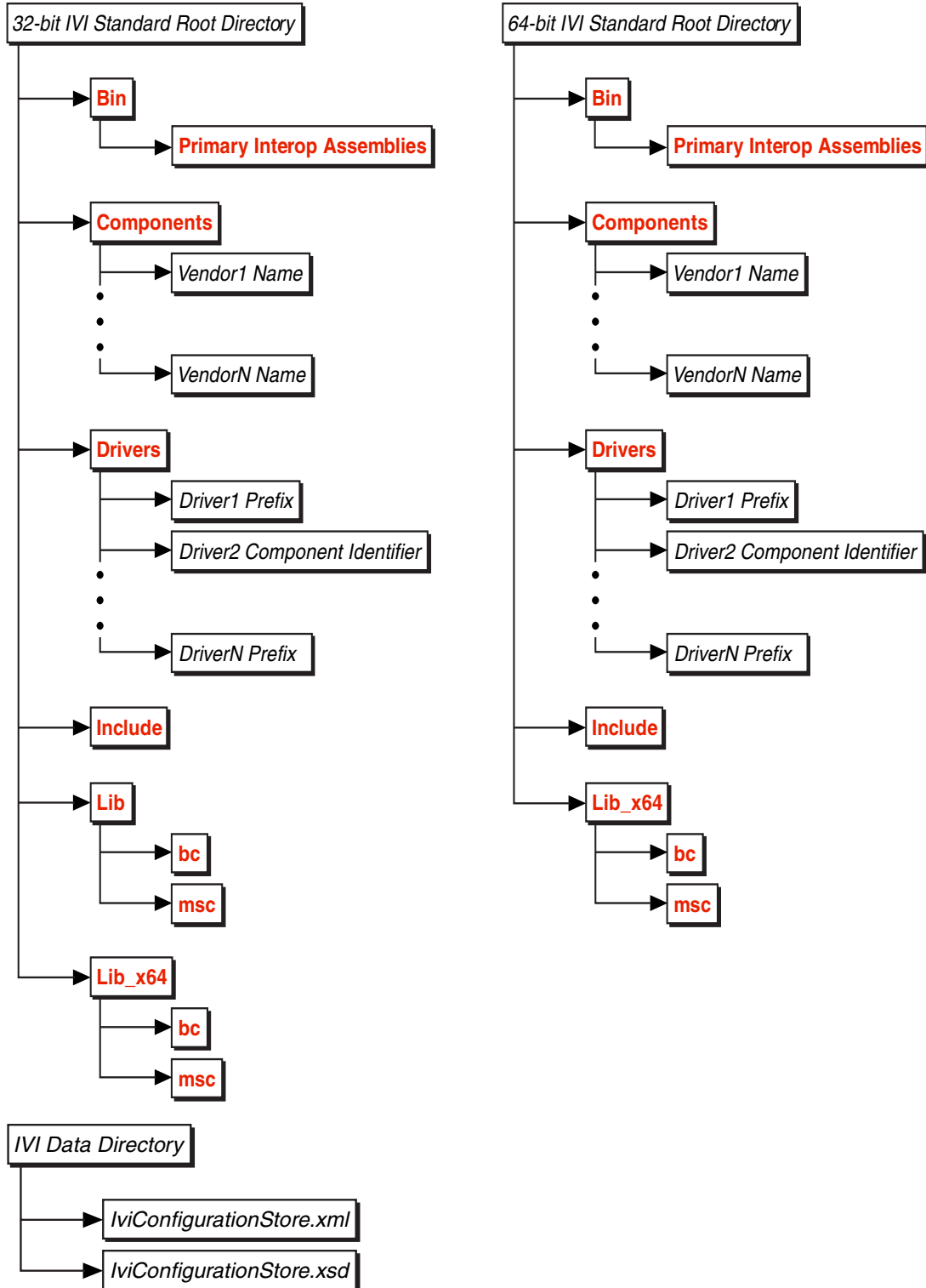
**Windows Vista 64**

C:\Program Files (x86)\ IVI Foundation\ IVI\ <i>(IVI Standard Root Directory)</i>	<i>32-bit Directory Structure</i>
C:\Program Files\ IVI Foundation\ IVI\ <i>(IVI Standard Root Directory)</i>	<i>64-bit Directory Structure</i>

Notice that the 32-bit IVI Standard Root Directory path on Vista 64 is different than it is on Windows 2000, Windows XP and Windows Vista 32. This is because on the 32-bit operating systems the 32-bit Windows Program Files directory name is “Program Files”, whereas on Vista 64 it is “Program Files (x86)”.

## 2.5.1 IVI 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.



## 2.5.2 IVI Standard Directory Tree

This section describes the IVI 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 shared components are also installed in the IVI standard directory tree. The root directory of the tree is called the IVI standard root directory.

### 2.5.2.1 Creation of the IVI Standard Directory Tree

The IVI shared component installer has the ability to detect whether the IVI 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 shared component installer, when called from another installer, has the ability to accept the standard root directory path as a command line parameter.

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

### 2.5.2.2 Contents of the IVI Standard Directory Tree

The root directory of the IVI standard directory tree is intended to contain only subdirectories. IVI 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 driver DLLs, all IVI shared component DLLs, and all vendor specific shared component DLLs. The `Bin` subdirectory contains the `Primary Interop Assemblies` subdirectory.

All IVI 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 shared component installer creates the `Bin` and `Bin\Primary Interop Assembly` subdirectories.

#### **Components Subdirectory**

The `Components` subdirectory contains the non-dispersed IVI shared component files and the non-dispersed vendor specific shared component files. The IVI 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 shared component installer creates the `Components` subdirectory.

Refer to Section 5.7, *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 shared component installer creates the `Drivers` subdirectory.

An IVI 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 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 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 shared component installer creates the `Lib_x64`, `Lib_x64\bc`, and `Lib_x64\msc` subdirectories.

## **2.5.3 Recommendations for Users**

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

- When installing a 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 driver, run the driver uninstaller.
- If you are no longer using IVI drivers and therefore do not need IVI shared components, use the Windows Add Remove Programs facility to remove the IVI shared components.
- If you are creating an installer that calls an IVI installer, refer to Section 7, *Installer Interface Requirements*, for details on command line syntax that IVI installers use.
- If you install the Microsoft .NET Framework after installing the IVI Shared Components and you intend to call an IVI-COM driver from a .NET language, run the batch file `IviPiaRegistration.bat` in the `<IVISTandardRootDir>\Bin\Primary Interop Assemblies` directory. On Vista 64, run this file in both the 32-bit and 64-bit directories.

## 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 error conditions that Sections 4, 5, and 5.8 specifically identify. In addition to these error 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 error condition occurs, the installer shall reverse the incomplete installation and exit.

If the installer is run in dialog mode and an error 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 error 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 error 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. Requirements for Creating and Detecting the IVI Directory Structure

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 Shared Component Installer Responsibilities

This section specifies the IVI 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 Shared Component Installer Responsibilities

The IVI 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. See Section 4.2, *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 4.2, *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 *VXPlug&play* directories. Refer to *VXPlug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the *VXPlug&play* directories.
  - **Windows Vista 64.** The 64-bit IVI standard root directory.
  - **Windows Vista 64.** 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 4.2, *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 Shared Component Installer Responsibilities

The 64-bit IVI 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. See Section 4.2, *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.
- 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 4.2, *Determining System Directories and Registry Keys*, for information on `<ProgramFilesDir>`.

- 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.
- 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 *VXIplug&play* directories. Refer to *VXIplug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the *VXIplug&play* directories.
  - The 32-bit IVI standard root directory.
  - The `<ProgramFilesDir32>` directory.
- The installer registers the 64-bit IVI standard directory path in the Windows registry as:  
`<HKLM\SW>\IVI, IviStandardRootDir`
- 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 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.

- The installer checks the Windows registry for a non-empty value of the following registry key:  
`<HKLM\SW>\IVI\CONFIGURATIONSERVER, MasterStore`
- 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 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.

- The installer adds `<IVISTandardRootDir64>\Bin` to the Windows system search path.
- The installer creates a Windows environment variable named `IVIROOTDIR64` and sets the value to be the `<IVISTandardRootDir64>` path.
- 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 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, <i>Driver Installer Responsibilities on 32-bit Operating Systems</i>
Singular 32-bit driver installer (64-bit OS)	4.1.2.2, <i>32-bit Driver Installer Responsibilities on 64-bit Operating Systems</i>
Singular 32-bit driver installer (32-bit OS/64-bit OS)	4.1.2.1, <i>Driver Installer Responsibilities on 32-bit Operating Systems</i> 4.1.2.2, <i>32-bit Driver Installer Responsibilities on 64-bit Operating Systems</i>
Singular 64-bit driver installer (64-bit OS)	4.1.2.3, <i>64-bit Driver Installer Responsibilities</i>
Unified 32-bit/64-bit driver installer	4.1.2.2, <i>32-bit Driver Installer Responsibilities on 64-bit Operating Systems</i> 4.1.2.3, <i>64-bit Driver Installer Responsibilities</i>

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

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

1. If the IVI 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 4.2, *Determining System Directories and Registry Keys*, for information on <ProgramFilesDir32>.
2. A failure condition exists if the IVI 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 VXiplug&play directory. Refer to VXiplug&play specification VPP-6: *Installation and Packaging Specification*, for details on the VXiplug&play directory.

Note: This behavior is required so that the IVI driver installer can pass a valid 32-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 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 4.2, *Determining System Directories and Registry Keys*, for information on <HKLM\SW>.

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

1. If the IVI 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 4.2, *Determining System Directories and Registry Keys*, for information on <ProgramFilesDir32>.

2. A failure condition exists if the IVI 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 *VXIplug&play* directories. Refer to *VXIplug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the *VXIplug&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 driver installer can pass a valid 32-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 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 4.2, *Determining System Directories and Registry Keys*, for information on <HKLM\SW>.

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

1. If the IVI 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 4.2, *Determining System Directories and Registry Keys*, for information on <ProgramFilesDir>.

2. A failure condition exists if the IVI 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 *VXIplug&play* directories. Refer to *VXIplug&play* specification *VPP-6: Installation and Packaging Specification*, for details on the *VXIplug&play* directories.
- The 32-bit IVI standard root directory.
- The <ProgramFilesDir32> directory.

Note: This behavior is required so that the IVI 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 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 4-2.** Windows identifiers for operating system directory paths

Directory / Registry Key	Windows 2000, Windows XP, and Vista 32	Vista 64
<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.

### **4.3 IVI Shared Component Installer Responsibilities on Windows Vista**

On Windows Vista 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. The installer shall set the attributes of the IVI standard root directories on Windows Vista to disable virtualization and allow modification without admin privileges. (This is an interim solution that will be reverted when the IVI Foundation specifies a revised directory structure that avoids placing writable files in the Program Files directory. Therefore, drivers and applications should not rely on users having write access to the IVI standard root directories.)

### **4.4 IVI Driver Installer Responsibilities on Windows Vista**

On Windows Vista, 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, *Requirements for Creating and Detecting the IVI Directory Structure*.

### 5.1 Driver Installation Procedure

The IVI driver installer program shall install driver files according to the following procedure:

1. The IVI 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. The IVI driver installer detects the IVI standard root directory as specified in Section 4.1.2, *IVI Driver Installer Responsibilities*.
3. For each supported operating system bitness, if the IVI standard root directory exists, the IVI driver installer checks for the presence and version of the IVI shared components as specified in Section 5.2, *Detecting the Presence and Version of the IVI Shared Components*.
4. For each supported operating system bitness, if the IVI standard root directory does not exist, or the IVI shared components are not installed or not of a sufficient version, the installer takes one of the following two actions:
  - a. The IVI driver installer calls the IVI shared component installer according to the requirements specified in Section 5.4, *Calling the IVI Shared Component Installer*. After the IVI shared component installation completes, the IVI driver installer repeats steps 1 and 2 to verify that the IVI shared component installer completed successfully.
  - b. The IVI 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 shared component installer and informs the user where to find the IVI shared component installer.
5. The IVI driver installer checks for the presence, vendor, and version of a previously installed IVI driver of the same name. The installer does this as specified in Section 5.3, *Detecting the Presence, Vendor, and Version of an IVI Driver*. If an IVI driver 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. 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 IVI driver installer may exit with a failure condition.
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 2.5.2.2, *Contents of the IVI 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 driver 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 driver 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 recreates the software module entry. Refer to Section 5.5, *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 Windows Add Remove Programs (ARP) facility.
13. If the installer is implemented with MSI technology, the installer shall not set the installed components to be “repaired” automatically.

## 5.2 Detecting the Presence and Version of the IVI Shared Components

An IVI driver installer shall determine the presence of the IVI 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 driver installer shall determine the version of the shared components by interrogating the value of `FileVersion` property of the `IVISharedComponentVersion.dll` file.

## 5.3 Detecting the Presence, Vendor, and Version of an IVI Driver

An IVI driver installer shall determine the presence of an IVI 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 driver installer may utilize a different detection mechanism if the implementation is functionally equivalent to the detection mechanism described in this section.

## 5.4 Calling the IVI Shared Component Installer

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

- For each supported operating system bitness, the IVI driver installer detects the IVI standard root directory according to the requirements specified in Section 4.1.2, *IVI 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 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 driver installer passes the user-specified directory path to the IVI shared component installer.
- If the IVI shared component installer causes the system to reboot after the IVI shared component installation completes, the IVI driver installer shall resume installation after the system has rebooted.
- The IVI driver installer verifies that the IVI shared component installer completed successfully by taking the following actions:
  - a) The IVI driver installer detects the IVI standard root directory as specified in Section 4.1.2, *IVI Driver Installer Responsibilities*.
  - b) If the IVI standard root directory exists, the IVI driver installer checks for the presence and version of the IVI shared components as specified in Section 5.2, *Detecting the Presence and Version of the IVI Shared Components*.
- If the IVI shared components are installed and are of sufficient version, the driver installer proceeds with driver installation.

## 5.5 Software Module Entries in the IVI Configuration Store

An IVI 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”.

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.

Note 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 Drivers

Package Type	Name	ModulePath32/ ModulePath64	Prefix	ProgID
IVI-C driver	Prefix	File name of software module	Prefix	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

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

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

### 5.5.1 Including Published API Collections in the IVI Configuration Store

An IVI 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-C or IVI-COM) 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

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

### 5.5.2 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.

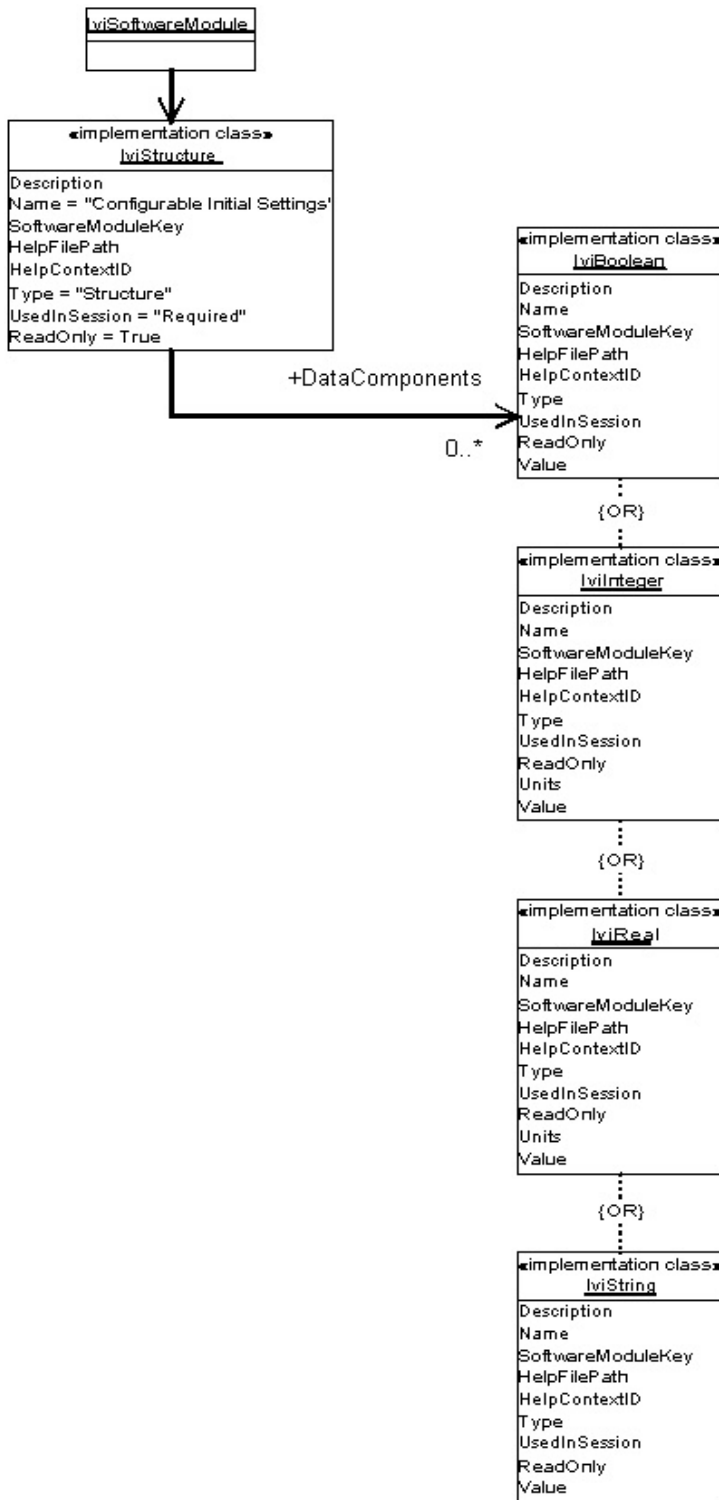
In cases where a repeated capability is defined by an IVI class specification, the RCName in the IVI Physical Name object shall be the repeated capability name as defined in the class specification.

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.

### 5.5.3 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 IVI 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 IVI driver has no configurable initial settings the installation program for the driver need not create the “Configurable Initial Settings” object.



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 IVI 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.

## 5.6 Driver Uninstaller

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

## 5.7 Installation of Vendor Specific Shared Components

An IVI 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 standard root directory tree, the installer shall comply with the following rules:

- The vendor specific shared component files shall be installed *after* the IVI shared components are installed. The installer detects the presence of the IVI shared components according to the requirements specified in Section 5.2, *Detecting the Presence and Version of the IVI Shared Components*. The installer may call the IVI shared component installer as specified in Section 5.4, *Calling the IVI 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 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 *VXIplug&play* specification *VPP-9: Instrument Vendor Abbreviations*. If the vendor specific subdirectory has not yet been created, the installer shall create it.

## 5.8 Installation of IVI Driver Start Menu Items

If an IVI 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 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.

## 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, *Requirements for Creating and Detecting the IVI Directory Structure*.

### 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

The IVI shared component installer shall install shared component files according to the following procedure:

1. The IVI shared component installer checks the bitness of the Windows operating system. A 32-bit IVI shared component installer exits with a failure condition if the operating system is not a 32-bit operating system. A 64-bit IVI shared component installer exits with a failure condition if the operating system is not a 64-bit operating system.
2. The IVI shared component installer detects and, if necessary, creates the IVI standard root directories according to the requirements specified in Section 4.1.1, *IVI Shared Component Installer Responsibilities*.
3. The IVI shared component installer checks for the presence and version of the IVI shared components as specified in Section 5.2, *Detecting the Presence and Version of the IVI 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, but 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. The IVI shared component installer upgrades a legacy IVI shared component installer if it exists on the system.
8. The IVI shared component installer installs the IVI Shared Component Cleanup Utility and registers it with the Windows Add Remove Programs (ARP) facility.
9. If necessary, the IVI shared component installer reboots the system.

## **6.4 IVI Shared Component Cleanup Utility Requirements**

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

In partial cleanup mode, the IVI 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 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 `IviConfigurationStore.xsd`.
- Deletes empty IVI standard common files directories.
- Deletes the IVI 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 Shared Component Cleanup Utility with the Windows Add Remove Programs (ARP) facility.

## 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

If the IVI 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]
```

If the IVI shared component installer is in Windows Executable (.exe) file format, it shall accept a command line argument of the following form and order:

```
IviSharedComponents.exe [IVISTANDARDROOTDIR=<path>] [IVISTANDARDROOTDIR64=<path>]  
[/q]
```

Optional arguments are enclosed in square brackets [ ]. Tokens are enclosed in angular brackets <>.

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

**Table 7-1.** Command Line Syntax

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 shared component installer shall ignore the <path> argument.

### 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. See Section 4.2, *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. See Section 4.2, *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. See Section 4.2, *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. See Section 4.2, *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.

## 9. Example Scenarios and Directories

Refer to *Appendix A: Example: IVI Driver Installer Scenarios*, for example scenarios of typical IVI driver installations. Refer to *Appendix B: Example: IVI Installation Directories*, for an example system directory on which a user has installed multiple drivers.

## Appendix A: Example: IVI 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 Vista 64, 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 Vista 64, 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 Vista 64, 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 Vista 64, 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 Installation Directories

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

Installation of a Fluke 45 IVI-C driver on Windows XP

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

Installation of a 32-bit and 64-bit Fluke 45 IVI-C driver on Vista 64

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

Installation of a 32-bit Rohde & Schwarz RsFs IVI-COM driver on Windows XP

Directory	Files	Comments
<b>C:\ProgramData\IVI Foundation\IVI</b>	IviConfigurationStore.xml	IVI data directory
	IviConfigurationStore.xsd	
<b>C:\Program Files\IVI Foundation\IVI</b>		32-bit IVI standard root directory
<b>\Drivers</b>		
<b>\RsFs</b>	<i>Source files, documentation, examples</i>	Driver specific files may be organized in subdirectories
	Compliance Document	
	Readme	
	RsFs.chi	
	RsFs.chm	
<b>\Bin</b>	RsFs.dll	32-bit DLLs
	<i>Other shared component dlls (e.g., IviFgenTypeLib.dll, IviConfigServerCAPI.dll)</i>	
<b>\Primary Interop Assemblies</b>	<i>.NET PIAs and corresponding XML IntelliSense help for IVI-COM shared component dlls (e.g., Ivi.Fgen.Interop.dll &amp; Ivi.Fgen.Interop.xml)</i>	32-bit PIAs
	Rs.RsFs.Interop.dll	
	Rs.RsFs.Interop.xml	
<b>\Include</b>	<i>Shared component include files (e.g., IviConfigServer.h)</i>	
	RsFs.h	
	RsFs_i.c	
<b>\Lib</b>		
<b>\msc</b>	<i>32-bit shared component import library files (e.g., IviConfigServer.lib)</i>	IVI-COM drivers do not normally have .lib files.
<b>\bc</b>		
<b>\Lib_x64</b>	<i>64-bit shared component import library files (e.g., IviConfigServer.lib)</i>	
<b>\msc</b>		

Installation of a 32-bit and 64-bit AgSAn IVI-COM driver with IVI-C Wrapper on Vista 64

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

Installation of a 32-bit and 64-bit AgSAn IVI-COM driver with IVI-C Wrapper on Vista 64 (Continued)

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