

## 7. Windows Platform-Based Segments

This chapter describes the technical requirements for building and integrating software components on top of the Windows platform. With the migration of the DII COE to the Windows 2000 family of operating systems the DII COE now supports both Windows 2000 and Windows NT. However DII COE compliance evaluations will only be performed on Windows 2000 platforms. Segments should install and be fully functional under both Windows 2000 and Windows NT. Any guidance specific to only one of the platforms will specifically state this fact. When the Windows platform is referenced in the remainder of this chapter, it is understood to mean both the Windows NT version 4 and Windows 2000 version 1 operating system families.

When the Windows Logo Program is referenced in the remainder of this chapter, it is understood to mean the program as defined by the requirements in the two application specifications:

Desktop - *Application Specification for Microsoft Windows 2000 for desktop applications.*

Server – *Application Specification for Microsoft Windows 2000, Server, Advanced Server, and DataCenter Server.*

These specifications will be referred to in the remainder of this chapter as the *Windows Logo Program Specifications*. The intent is to use the Windows Logo Program as the baseline set of requirements that Win32 and Win64 software components for COE platforms must meet. The main goals of the COE and Windows Logo Program are the same.

- **“Tested”** - The segment has been tested and is fully functional on the Windows 2000 platforms.
- **“Best User Experience”** - The segment has been designed to provide optimum usability and to ensure a consistent, accessible user experience.
- **“Works well with Other Applications and Devices”** - Segments are designed to take advantage of the latest software & hardware technologies available on the Windows platforms.

It is recognized that the COE is defining a specific environment in DoD to ensure security, interoperability and peaceful coexistence, which goes beyond those needed for general use in the commercial market place. In recognition of these unique requirements, this chapter and the developers’ guidance section in chapter 9 are designed to provide guidance for:

- Unique DII COE requirements that are not addressed in the *Windows Logo Program Specifications*,
- DII COE requirements that are different from those directed in the *Windows Logo Program Specifications*, and

- Legacy segments built to previous versions of the DII COE and I&RTS.

This chapter also provides guidance in areas where there are important differences between Windows and UNIX. The COE concepts are not specific to UNIX, or Windows, or any other operating system or windowing environment. However, certain adjustments to COE implementation details are required to support differences between the Windows and UNIX environments, and to meet the additional technical requirements stipulated by the Windows environment (e.g., registry, Active Directory).

All of the requirements discussed in other chapters, except as noted in those chapters, are also requirements for Windows segments. Chapter 6 in particular describes the segmentation process and the format of segment descriptors that are used for both UNIX and Windows.

The extensions described in this chapter to accommodate Windows are not platform-dependent. The DII COE will support Windows NT on:

1. Windows NT Workstation;
2. Windows NT Server; and
3. Windows NT Server, Enterprise Edition

And Windows 2000 will be supported on:

1. Windows 2000 Professional (Workstation);
2. Windows 2000 Server; and
3. Windows 2000 Advanced Server

Of these, the DII COE has only been tested on the first 3 configurations for Intel 80x86 derivative hardware. The Datacenter Server and Appliance Kit versions of Windows 2000 are unique to each vendors hardware configuration. Therefore, the only supported configurations are the three Windows 2000 products listed above.

The Kernel Platform Configuration program (see Appendix A) provides a process whereby programs may submit other Windows operating system versions or hardware configurations for validation.

### 7.1 Windows COE Architecture

UNIX and Windows architectures are similar in many ways, but there are also many fundamental differences. A goal of the COE is to capitalize on the similarity and to negate or minimize the impact of the differences.

Figure 7-1 **Error! Reference source not found.** is a simplified diagram that illustrates the relationships between the Windows operating systems internal components and the COE layers. The labeled boxes in the Infrastructure Services and Common Support Applications layers are not intended to be all encompassing nor are they intended to conflict with the COE architecture diagrams in Chapter 2. The boxes are representative of the services and COE-component segments in the COE and is designed to illustrate the COE platform architecture, as defined in Chapter 2, from the view of an Windows COE based platform. The Infrastructure Services, Common Support Applications, and mission-application segments may reside on one platform or may be distributed on separate servers and workstations. **Error! Reference source not found.** Figure 7-1 is provided merely as a common reference point for discussions of the Windows architecture and the COE in the sections that follow.

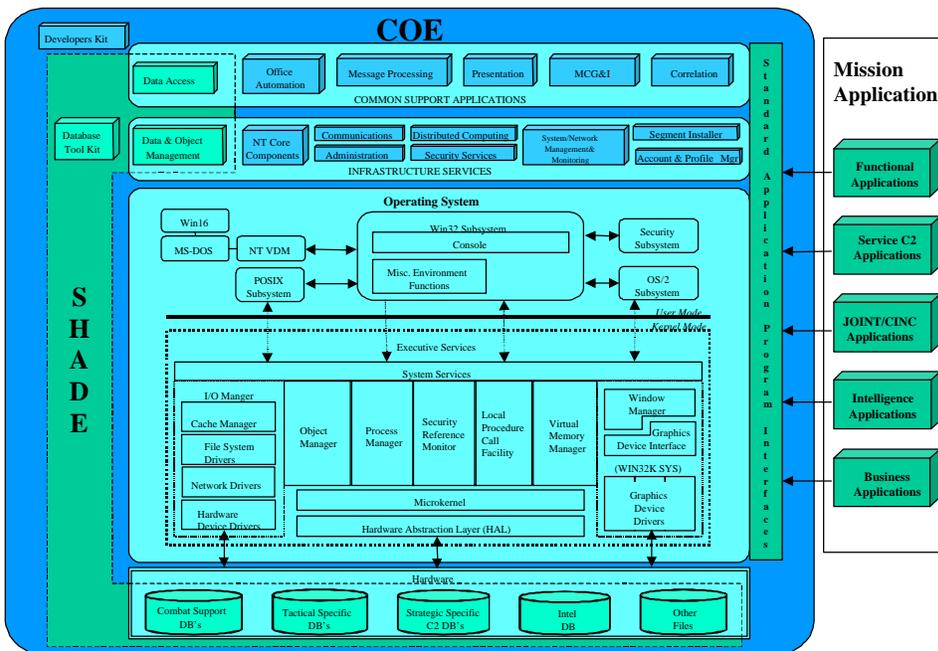


Figure 7-1: COE Architecture on the Windows Operating Systems

## 7.2 Windows COE Segmentation

Chapter 6 describes the segmentation process in detail. The segmentation process is primarily aimed at providing:

- a standardized approach for software installation and uninstall,
- for providing a method for software to state dependencies and conflicts,
- and to allow software to describe system resources it needs so that the operating system and COE can arbitrate potential conflicts.

However, many COTS products are already packaged with their own method for doing software installation and it is not always cost-effective to “fully segment” such products. In consideration of this fact, Chapter 6 also describes an “abbreviated segmentation” process that allows products, particularly COTS products, to be installed through vendor-provided instructions and yet preserve the benefits of the COE segmentation concept. Abbreviated segmentation is even more important in the Windows environment due to the much larger software base of already-developed applications that will be useful in a COE-based system.

- **Full Segmentation:** Full segmentation is the incorporation of segment descriptor files (SDFs) with executables, icons, menus, and data into a complete software package that is installed using the COE-provided segment installer, `COEInstaller`. Segments shall be location insensitive so that they may be installed in any directory chosen by the operator.
- **Abbreviated Segmentation:** Abbreviated segmentation is the incorporation of the DII COE required SDFs into an installation package that works through the `COEInstaller` with another, Windows Logo Program compliant, installation service to install the software or data.

Subsections 7.2.1 and 7.2.2 below discuss segmentation issues that are important for the Windows environment for both “full” and “abbreviated” segmentation.

The degree to which “plug and play” between segments is possible is highly dependent upon the degree to which segments are Windows Logo Program-compliant. For this reason, the *I&RTS* fully supports the Microsoft Logo Specification approach as a subset of the requirements for full DII compliance regardless of whether full or abbreviated segmentation is chosen. DII compliance for Windows segments measures the degree to which a segment or system achieves conformance with the rules, standards, and specifications identified by the COE and the Windows Logo Program. Segments shall use the *Windows Logo Program Specifications* for development guidance and shall meet Windows Logo Program specifications with the exception of the items described in section 9.6.1. Segments are not required to have a Microsoft Logo license.

## 7.2.1 Full and Abbreviated Segmentation Requirements

The requirements listed below apply equally to full and abbreviated segments, except as noted.

- Developers may choose either full or abbreviated segmentation for Windows COTS products, but shall obtain prior Chief Engineer approval for selecting the abbreviated segmentation process.
- Developers shall use full segmentation for GOTS segments.
- Both full and abbreviated segmentation requires that SDFs be created in accordance with Chapter 6 and the appropriate sections below. As per Chapter 6, segment descriptor information is contained in files located under *SegDir*\SegDescrip whether full or abbreviated segmentation is used. (*SegDir* is the segment's assigned<sup>1</sup> directory, **not** the segment prefix although they maybe the spelled the same.) As per the requirements that follow, the location of *SegDir* will be determined at installation time.
- DII compliance requires that developers create segments that are location insensitive.
- Segments shall operate under Windows 2000 in accordance with the Windows Logo Program specifications, except as specified in section 9.6.1.
- At a minimum all segments shall properly register the segment and components in accordance with the *Windows Logo Program Specifications*. Segments using the full segmentation process may use the `PostInstall` segment descriptor to properly register the segment components with the operating systems registry.
- Segments shall completely uninstall from a system by means of an automated, registered uninstaller<sup>2</sup>. Segments shall not remove core components or shared components necessary to other applications.
- Segments that contain hardware device drivers shall at a minimum support Windows 2000; device drivers for Windows NT/95/98 are optional. Device drivers shall be compliant with the Windows Logo Program for Hardware. Hardware and device

---

<sup>1</sup> Segments sometimes use the segment prefix as the name of the assigned directory, which is permissible as long as the segment prefix is unique among all segments. This is not always the case, as explained in Chapter 6, because developers sometimes need to share segment prefixes among segments, such as in an aggregate segment. Therefore, to guarantee uniqueness, the assigned directory is used rather than the segment prefix.

<sup>2</sup> Segments that use the `COEInstaller` are deemed to meet this requirement since it is the tool's responsibility to be Logo-compliant. Segments that use abbreviated segmentation must ensure that the vendor-provided installation process meets this requirement.

drivers shall follow the applicable design guide<sup>3</sup>: *PC 99 Hardware Test Specification 2.0 Release*.

## 7.2.2 Abbreviated Segmentation Considerations

This section discusses additional considerations appropriate for abbreviated segmentation, which is especially relevant to products that are installed using a Microsoft Software Installation (MSI) Windows installation package rather than the COE Segment Installer tool. The abbreviated segment shall be installable out-of-the-box and even though a commercial installation program is used for the installation, it is necessary to provide some segment descriptor information for use by the COE Installer. In particular, the COE needs such information so that it can recognize the existence of an abbreviated segment, so that other segments can state dependencies on the abbreviated segment, and to be aware of certain characteristics of the segment, such as disk space used.

- The cognizant Chief Engineer must approve the installation methodology and sequencing of the COTS installation package and the segment descriptor files.
- To minimize installation and operational problems with filename and environment variable conflicts, all segments shall comply with the Windows Logo Program default installation location. The default installation location is found by querying the following registry key:

`HKLM\Microsoft\Windows\CurrentVersion\ProgramFilesDir.`

The value of this key is established when the kernel is installed, and for DII-compliant platforms, the registry key value will be `drive:\Program Files`. The installation program shall query this key to ensure that segments load correctly if, for example, a user has renamed this directory on their machine.

- The installation program, except for COTS, shall check up-front whether the current user is a member of the administrator's local group; when not a member the Installer shall fail gracefully.
- In accordance with the Windows Logo Program, the Installer shall fail gracefully in the case where a file cannot be installed or replaced because file system security prevents an existing file from being written or overwritten. This will avoid confusion later by preventing the user from getting file copy errors from which there is no recovery.
- An abbreviated segment requires a `DEINSTALL` segment descriptor file unless it is a "permanent" segment as described in Chapter 6. This descriptor file, however, for abbreviated segmentation will typically not have any executable statements in it. It serves only to let the COE installer know that it is permissible for a user to delete the segment. The actual segment removal is performed by the vendor-provided uninstall

---

<sup>3</sup> Documents are available at <http://www.microsoft.com/hwtest/systems/>

program accessed through the Add/Remove Programs applet in the Control Panel. If the actual product is removed, the COE Segment Installer tool shall be used to remove<sup>4</sup> the accompanying segment descriptors.

- State in the ReleaseNotes segment descriptor that the abbreviated segmentation process has been used to create this segment.
- The disk space requirements listed in the Hardware descriptor shall be the sum of the space required for the segment descriptors and for the software loaded through vendor-provided procedures. (change to follow installation process of which goes first\*\*\*\*\*)

Deleted: \*  
Deleted: \*\*\*\*\*

### 7.2.2.1 Windows Installer

With the adoption by the COE of the *Application Specification for Microsoft Windows 2000 for desktop applications* the COE requires desktop segment installation packages to use the MSI technology. The MSI technology is divided into two parts that work in combination: A client-side installer service (Msiexec.exe), normally referred to as the Windows Installer, and a MSI package file. The Windows Installer uses information contained in the MSI package file to install the program.

The Msiexec.exe program is a component of Windows Installer. When it is called by Setup, Msiexec.exe uses Msi.dll to read the package (.msi) files, apply any transform (.mst) files, and incorporate command-line options supplied by Setup. The installer performs all installation-related tasks, including copying files onto the hard disk, making registry modifications, creating shortcuts on the desktop, and displaying dialog boxes to prompt for user installation preferences when necessary. Each MSI package file contains a relational-type database that stores instructions and data required to install/uninstall the program across many installation scenarios.

The Windows Installer is not just an installation service; it is an extensible software management system. Windows Installer manages the installation of software, manages the additions and deletions of software components, monitors file resiliency, and maintains basic disaster recovery by using rollbacks.

The Windows Installer features include:

- Restores original computer state upon installation failure: Windows Installer keeps track of all changes made to the system during the program installation process. If the installation does not succeed, the installer can restore the system to its initial state. This is known as "rollback."

<sup>4</sup> This is a near-term implementation restriction. Transition to a commercial installation process will integrate SDFs so that they are automatically removed during deinstallation without the need to take an explicit separate step.

- Ability to set file and directory security permissions.
- Helps prevent certain forms of inter-program conflicts: It is not unusual for a program that is being installed or uninstalled to cause problems with another program already on the computer, or even to cause the computer to stop responding (hang). The installer enforces installation rules that help prevent conflicts caused when an installation operation makes updates to a dynamic-link library (DLL) file shared by an existing program, or when an un-installation operation deletes a DLL file shared by another program.
- Diagnoses and repairs corrupted programs: A program can ask the installer to determine whether an installed program has any missing or corrupted files. It can then ask the service to repair that program as necessary by copying again only those files found to be missing or corrupted.
- Reliably uninstalls existing programs: The installer can reliably uninstall any program it previously installed, removing all the associated registry entries and program files, except for those shared by other installed software.
- Supports the on-demand installation of program features: The installer can be instructed to initially install a minimal subset of a program. Later, additional components can be automatically installed the first time you use a features that require additional components.
- Supports unattended program installation: The installer supports the ability to script a program installation according to administrator instructions.

### **7.2.2.2 Windows Installer Requirements**

Many Windows-based software segments are built using the abbreviated segmentation process. This is done to take advantage of the ability to use the same version of the segment for COE and non-COE systems. The Windows Installer supports the ability to meet the COE's and Logo Programs install and uninstall requirements. This section provides the additional requirements for GOTS abbreviated segment Windows installer (MSI) packages not addressed in the Logo Program specifications. COTS Windows Installer packages that do not meet these requirements should be extended to include these requirements by the SSA or integrating agency.

- The installation package shall include a copy of the Windows NT version of the Windows Installer service installation program. The included Setup.exe file shall check to see if the operating system is Windows NT and if it is and the Windows Installer service is not installed, shall then install the Windows Installer service or direct the individual to install the Windows Installer before proceeding.

- The installation package at start up of local installs must check to see if the individual performing the installation is a member of the Administrators group. If the individual is not a member of the Administrators group, the installation package shall present a message that administrative privileges are required to complete the installation and then gracefully fail.
- Ensure the costing action is initiated. Costing is the method used by the Windows Installer to determine the disk space required by all the components in an installation. Costing takes into account factors such as whether existing files need to be overwritten.
- The MSI file shall include a screen or screens, at or near the beginning of the installation sequence, that perform at a minimum the following:
  - a. Ability to display the segments `ReleaseNotes` e.g., Figure 7-2: COE Segment Welcome Screen.
  - b. Ability to display the segments software and/or hardware conflicts e.g., Figure 7-2: COE Segment Welcome Screen.
  - c. Ability to display the segments software and/or hardware dependencies e.g., Figure 7-2: COE Segment Welcome Screen.
  - d. Ability to display a Help screen, which details the installation procedures.
  - e. A `Next` button event sequence that validates any dependency and/or conflict before displaying the next screen. If a dependency or conflict exists that will not allow the segment to function, the installation package does not advance to the next screen and a warning message is presented explaining what action must be taken before the segment can be installed. See Figure 7-5: Dependency Warning Screen, for an example of a warning screen. If the segment will function but due to the dependency or conflict its functionality will be limited. The installation package may elect to display a warning message explaining the impact on the segment and allow the individual installing the segment to continue or cancel the installation.
  - f. Display the segments name, prefix, and version number e.g., Figure 7-2: COE Segment Welcome Screen



Figure 7-2: COE Segment Welcome Screen

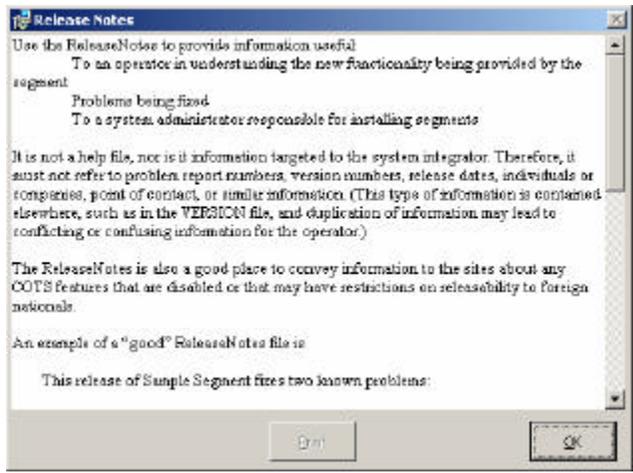
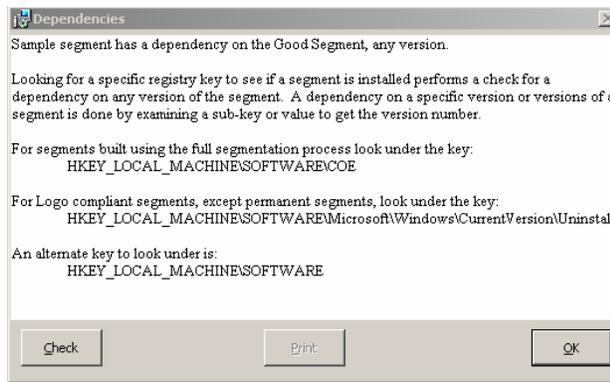
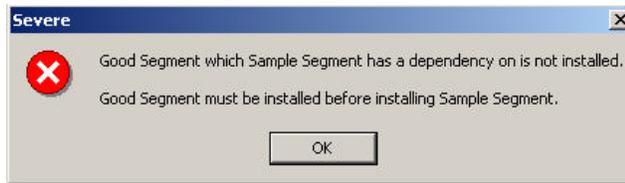


Figure 7-3: ReleaseNotes Screen



**Figure 7-4: Dependency Screen**



**Figure 7-5: Dependency Warning Screen**

- Provide a batch or command file that can be used to initiate a silent/unattended installation of the segment.
- Windows Installer installation packages that perform as a Patches or Upgrade are required to have a corresponding set of segment descriptor files.
- The segments entry in the *Add/Remove Programs* applet shall contain a support information line that when selected displays information on the segment as illustrated in Figure 7-6. The registry entries required for this screen are detailed in section 9.6.6.

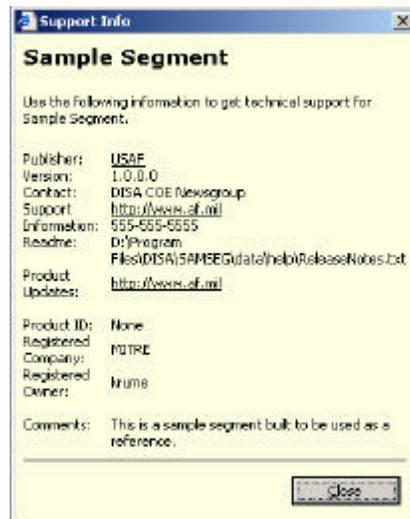


Figure 7-6: Add/Remove Programs applet Support Information

### 7.3 Segment Descriptor Considerations for Windows

Chapter 6 describes the segment descriptors required for segments whether they are Windows or UNIX, and whether they are a result of full or abbreviated segmentation. This section is provided as a quick reference for items that are Windows-related. Refer to Chapter 6 for complete discussion of each of the descriptors discussed below.

General comments follow:

- Pathnames shall be given using ‘\’ in conformance to the Windows environment.
- Filenames and directory names are to be considered as if they are case sensitive, even though Windows itself is not case sensitive. This will minimize porting efforts between Windows and UNIX platforms.
- Segments should normally not need to specify a disk drive because such designations are considered to be advisory only. For backward compatibility, when a disk drive designation is given, it and any associated pathname must be enclosed in double quotes. This is required so that the tools can distinguish between use of ‘:’ as a field delimiter for descriptor lines, or as a separator between a disk drive name and a directory pathname.
- In accordance with commercial standards, executable descriptors shall have either a .EXE extension (for compiled programs) or a .BAT/.CMD extension (for batch files). This applies to the “scripts” used in the installation process: DEINSTALL,

PostInstall, PreInstall, and PreMakeInst. Segment descriptor files may optionally have a .TXT extension.

Comments related to specific descriptors follow.

### **COEServices**

#### **DEINSTALL.EXE and DEINSTALL.BAT**

A segment that does not include a DEINSTALL descriptor is deemed a “permanent” segment and may be updated, but not removed. In many situations, it is desirable for the segment to be removable, but there are no actions that DEINSTALL must perform. For this reason, DEINSTALL may exist as a zero-length file and it may exist as a file with no extension.

### **Direct**

Segments should normally not register information in the Windows NT Control Panel Add/Remove Programs application because the segment installer will perform this function in accordance with the Windows Logo Program requirements. However, abbreviated segmentation segments that use a Logo-compliant installation process may already do this.

### **FileAttribs**

The FileAttribs descriptor is not presently supported for Windows platforms.

### **Hardware**

The *diskname* field for the \$PARTITION keyword must be a disk drive name. For example, to indicate that a segment requires 20MB on the F disk drive, the proper \$PARTITION statement is:

```
$PARTITION:"F:" :20480
```

### **Network**

The Network descriptor is not presently supported for Windows NT and Windows 2000 platforms. VerifySeg will issue a warning if a Network descriptor is found for an Windows-based segment.

### **Processes**

The \$RUN\_ONCE keyword identifies process that should be run the next time the system is started. This keyword requires approval by the cognizant DOD Chief Engineer because of potential security and performance risks.

The *username* field for \$BOOT is ignored for Windows. On Windows platforms, \$BOOT, \$RUN\_ONCE, and \$PERIODIC processes run as a service and are thus run as a system user. They cannot be run as an arbitrary user.

### **Registry**

The Registry descriptor allows the segment to have the COEInstaller create registry key entries.

## 7.4 Platform Configurations

All PC hardware shall be Windows 2000-compliant, and services and agencies are required to select hardware platforms that are contained on the Microsoft *Hardware Compatibility List*. Programs that do not follow this requirement should not expect to receive assistance from the DII COE Engineering Office concerning hardware compatibility problems.

Windows platforms may be interconnected into Windows NT domains and workgroups or into Windows 2000 Active Directory domains, trees, or forest. Windows platforms, depending upon how they are to be used in the architecture and what operating system software version is loaded on them, may be further classified as either a server or workstation<sup>5</sup> configuration.

Core components can be installed on top of the operating system, expanding the systems capabilities and functionality. Core components are defined as components purchased with the commercial Windows operating system and are installable under the same license. Utilities and files included in Windows Service packs by definition become core components upon installation. Examples of core components include:

- Chat
- Clock
- Domain Controller (server only)
- DNS Server (server only)
- Internet Explorer
- Public-key (PK) cryptographic services
- Internet Information Server (IIS) (server only)
- Terminal Server (server only)
- Active Directory (server only)
- Certificate Services (server only)

Core components provided by the vendor do not have to be packaged in segment format. They may be installed using vendor-provided instructions as part of loading the DII COE kernel, or if the vendor instructions permit it, after the operating system and kernel have been loaded. Being a Windows operating system core component does not mean the DII COE endorses their use.

### Security Considerations

See Chapter 4 for Windows platform security considerations.

---

<sup>5</sup> As noted in section 7, the COE has only been tested on Intel 80x86-derivative hardware for Windows 2000 Professional, Server and Advances Server Edition configurations.

### 7.4.1 General Directory Structure Requirements

The basic directory structure described in Chapter 6 is supported on Windows platforms for legacy segments. Chapter 6 shows the root directory for segments as \h. However, the practice on Windows platforms is to use the standard directory \Program Files as the root directory for applications. Therefore, the COE tools are migrating to this practice. In the interim, Windows COE platforms may have both the legacy \h and the \Program Files convention. The intent is to ensure that the DII COE directory structure follows commercial practices, when practical, and is thus compliant with the Windows Logo Program.

The present COE installation software<sup>6</sup> will attempt to put segments on the primary disk drive first. However, an operator should override this in order to conform to conventions described in the Site Integrators Guide. Segments should only be installed on the operating system drive (partition) when disk space on other logical drives is unavailable.

The following guidance is provided for Windows-based segments.

- The directory structure for Windows segments shall be self-contained and shall be location- insensitive. This then allows the decision of where to place a segment to be made at installation time and not a hardcoded requirement of the segment. Segments shall use the Win32 API, in accordance with the Windows Logo Program, to return handles to directory locations rather than relying upon a fixed, hardcoded structure.
- The segment directory structure shall be implemented in accordance with the *Windows Logo Program Specifications*. This means that segments will be located by default underneath the standard directory \Program Files\SegDir. But as per the previous requirement, an operator will choose whether the segment is to be installed as per industry convention (i.e., \Program Files) or as per local site conventions.
- Segments that share files shall locate shared files underneath the standard directory

\Program Files\Common Files\SegDir Shared

where *SegDir* is the “owning” segment’s assigned directory. The default common files root directory is found by querying the below registry key:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\CommonFilesDir.

Subdirectories underneath this Shared subdirectory should be named to correspond to the file type for the shared files. For example,

\Program Files\Common Files\SegDir Shared\DLL

---

<sup>6</sup> A future release of the COEInstaller will be provided to conform to the conventions for disk partitions described in the Systems Integrators Guide.

would be appropriate for shared DLLs.

- Developers shall place segment descriptor information in the subdirectory `\h\SegDir\SegDescrip` as described in Chapter 6.
- Segment data files that are dynamically updated as a user interacts with the segment, but that are identified with a user role and not a specific user, shall be placed underneath one of the following standard directories, depending upon the data scope:

Data drive:\data\local\SegDir\data	(local scope)
Data drive:\data\global\SegDir\data	(global scope)
Segment drive:\Program Files\SegDir\data	(segment scope)

where *SegDir* is the segment's assigned directory. Files that contain data support functions specific to the data files may also be located in the above `\data` directories. The supporting files must be specific to the data files. An example of this is a DLL that contains routines to extract information from the data files specific to each type of aircraft. The DLL embedded routines only work with that specific aircraft's data files and the mission-application segment can have multiple aircraft data segments installed, based on mission requirements, each with its matching DLL. The intent is to support the ability to build a data segment that contains the data and supporting files together into one deployment package.

- The `Personal` subdirectory shall be used to store operator preferences (rather than the UNIX `Prefs` subdirectory described in Chapter 6).
- Refer to subsections 7.4.2 and 7.4.3 below for detailed guidance on conventions for global and group data requirements.
- Refer to subsection 7.4.3 below for guidance on directory structure naming conventions for subdirectories underneath *SegDir*.
- Developers shall ensure that the segment separates the runtime environment from the development environment. This requirement is so that a segment delivered to a site will contain only those files required for operation and will not include source code, makefiles, header files, etc. It also simplifies configuration management so that segment files can be collected as appropriate for operational sites (executables and data). Segments providing public APIs shall include header files and linkable libraries as part of the runtime environment and development environment. Refer to Chapter 9 for implementation details.
- Developers shall ensure that classified files are separated from unclassified files and delivered in separate segments as described in Chapter 6.

## 7.4.2 Segment Subdirectory Structure

The Windows Logo Program does not provide guidance for subdirectories below a segment's home directory. The intent here is to provide guidance on the subdirectory structure for all types of segments, except COTS. This guidance builds on Chapter 6 and the Windows Logo Program directory structure requirements.

The intent is to have segments create only those subdirectories required to store files. Empty subdirectories should be created during the segment's installation when files will be created in the subdirectory by the segment. Subdirectories other than those listed below may be created as required, but such directories shall not duplicate the purpose and function of any required subdirectory. Subdirectory names, for new subdirectories, shall meet the naming conventions in subsection 7.6.1.

Except COTS, segments shall use the following subdirectories under *SegDir* for files that fit within the categories identified:

- `\bin` - executables and DLLs that are not shared
- `\data` - static data, should only contain subdirectories or individual data and support files
- `\data\INI` - initialization (`.ini`) files
- `\data\Help` - help (`.hlp`) files
- `\Doc` - documents and manuals (help files do not go in this subdirectory)
- `\Scripts` - interpreted code files (e.g., `.bat`, `.cmd`, `.pl`)
- `\Setup` - setup files used to reconfigure<sup>7</sup> the segment or Suite after installation.

Except COTS, segments shall use the following subdirectories underneath their `\Program Files\Common Files\SegDir Shared` for files that fit within the categories identified:

- `\data` - shared static data files
- `\DLL` - shared library files.

Directory and subdirectory requirements for segment submission to the SSA are discussed in subsection 9.6.16.

Figure 7-7 **Error! Reference source not found.** is a sample segment drive that illustrates the following key points about segment subdirectory structures:

- The Netscape web browser, a COE-component segment, is installed out-of-the-box into the default home directory `\Program Files\Netscape`.

---

<sup>7</sup> Do not confuse the `Setup` subdirectory with the `SegDescrip` subdirectory. The `SegDescrip` subdirectory contains information about segment installation. The `Setup` subdirectory contains files that are used to reconfigure a segment after installation has already been performed, perhaps even after a segment has been operational for a while and needs to be restored to the state was in after installation.

- The abbreviated software segment GBSSBM, a mission-application segment in the GBS Suite, has two separate directory structures. First, the segment's home directory \Program Files\GBS\GBSSBM is created during the segment's installation routine. The segment's commercially based installation routine is developed to install the segment into this directory, in accordance with the Windows Logo Program. Second, the segment's COE required directory, \h\COTS\GBSSBM, and subdirectories are created by the COEInstaller. Subdirectories are created based on the segment's segment descriptor files.

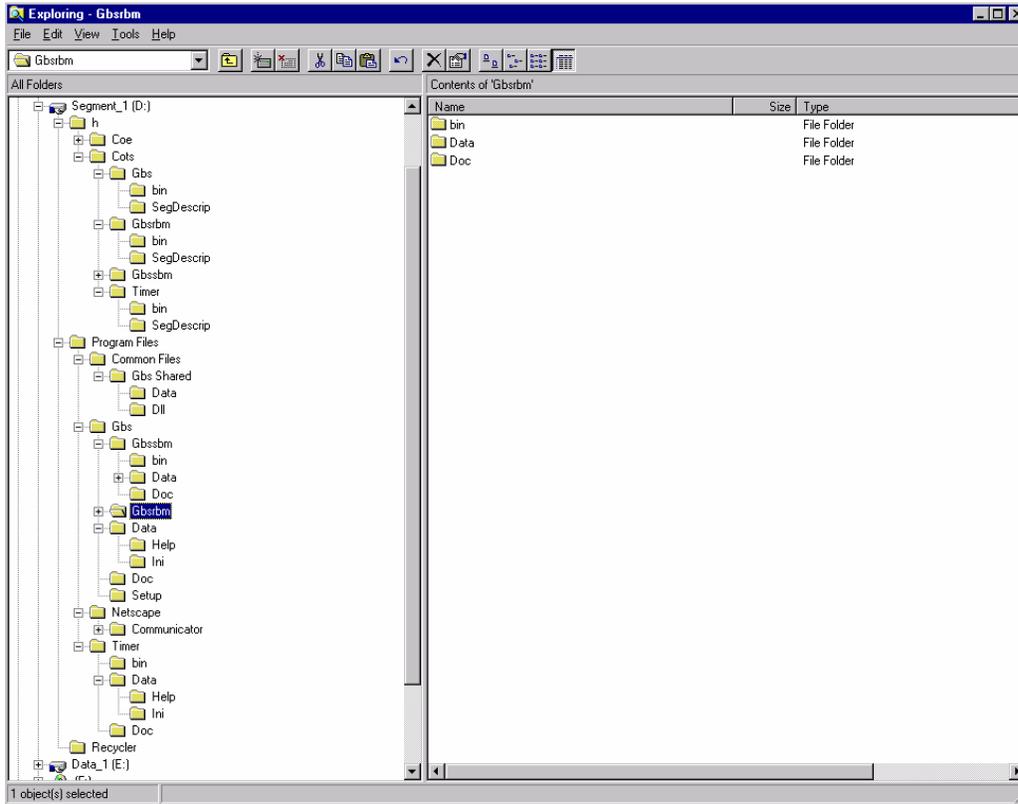


Figure 7-7: Segment Directory Structure Example

### 7.4.3 Data Directory Structure

Figure 7-8 [Error! Reference source not found.](#) is a sample data drive that illustrates the following key points about directory structures for dynamic segment data:

- Chapters 5 and 6 define data in terms of data scope. The intent of this section is to show how the guidance in Chapters 5 and 6 on data, with local or global scope, is applied to Windows-based COE platforms. The data directory structure is the same for all Windows-based COE platforms, whether a workstation or server. Static data used by a segment shall be stored in the \data subdirectory under the segment's home directory, as shown in **Error! Reference source not found.**
- Data that is dynamic in nature shall be stored on a logical drive designated as a data drive. Data that is managed and updated by the segment during operations, where the segment and data reside on the same platform (local scope), shall be stored in the directory Data drive:\data\local\SegDir\data, where SegDir is the segment's assigned directory. An example of this structure is the TIMER segment shown in **Error! Reference source not found.**
- Dynamic segment data with global scope shall be stored in the directory Data drive:\data\global\SegDir\data. An example of this structure is the GBSSBM segment shown in **Error! Reference source not found.** Data stored in the subdirectory Data drive:\data\global\GBSSBM\data is created, updated, and deleted by GBSSBM client systems. Normally the directory is shared on the network to allow client access to the data.

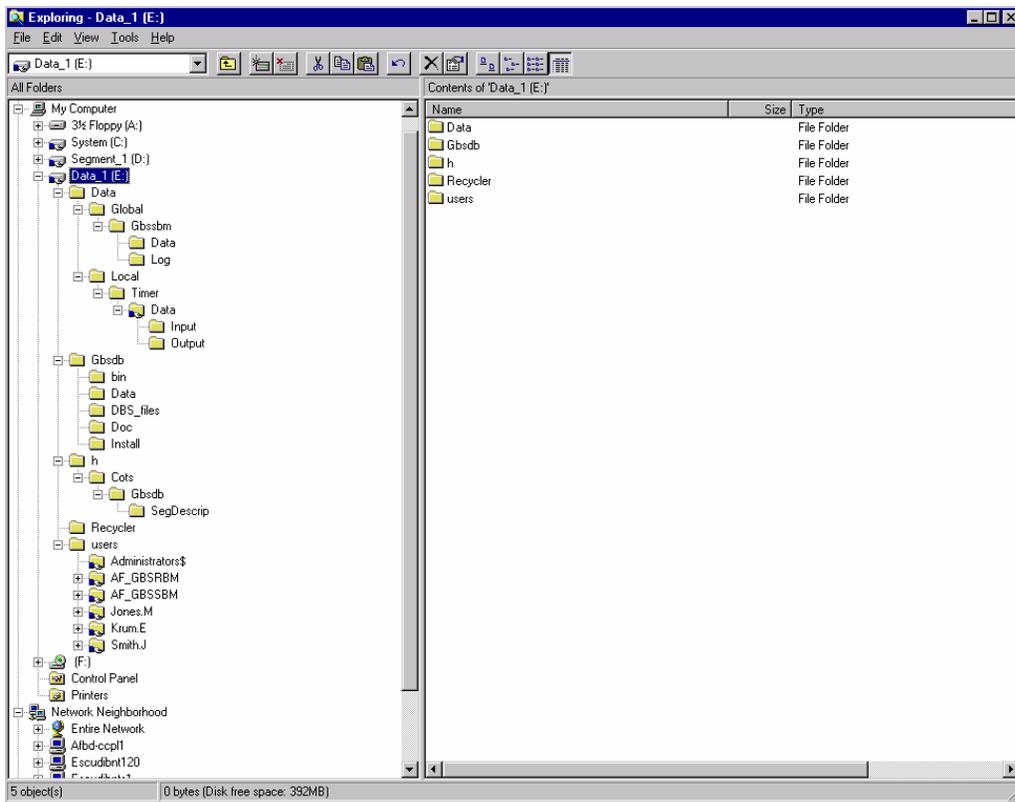


Figure 7-8: Data Directory Structure Example

- Data managed by a database management system like Oracle, Sybase, Informix, or SQL Server is stored on a database server underneath the directory `Data drive:\SegDir\DBS_files`. Database data files (tablespace, index, log) may be located on the same drive (partition) with the `\USERS` directory, or it may be on one or more dedicated physical or logical drives. **Error! Reference source not found.** illustrates a combined users and data drive. The database segment directory shown has a segment whose assigned directory name is `GBSDB`. The subdirectory structure underneath this segment is dependent upon the segment's design. The data access standards (JDBC, ODBC, OLE-DB) used between client and database server, or server and database server, do not change the COE directory structure requirements. If the segment supports more than one mission application, each mission application's data files shall be in a separate subdirectory. For example,

`Data drive:\GBSDB\DBS_files\MissionApp.`

Every Windows-based database segment, when required shall have the following subdirectories under their home directory:

- \bin - executables and DLLs that are not shared
- \data - static data, should only contain subdirectories or individual data and support files
- \data\Help - help (.hlp) files
- \DBS\_files - database files
- \Doc - documents and manuals (help files do not go in this subdirectory)
- \install - scripts to install and create the database segment.

## **7.5 Reserved Words**

The Windows Logo Program refers to reserved words as “Keywords.” Keywords consist of:

- segment prefixes,
- environment variables,
- reserved filenames, and
- reserved directory names (includes the complete path).

Windows operating system files are called *Core files* and are reserved as keywords.

### **7.5.1 Segment Prefixes**

The segment prefixes listed as reserved in Chapter 6 are also reserved in the Windows-based COE. A current list of segment prefixes is available from the COE Engineering Office. The following are reserved segment prefixes specific to the Windows-based COE:

NT  
WIN  
WIN95  
WIN98  
WIN2K  
WINNT  
WinXP.

### **7.5.2 Environment Variables**

Historically, the COE environment variables are based on the UNIX operating system’s requirements. Windows was developed after UNIX and, as a result, has different names for equivalent UNIX environment variables. The environment variables<sup>8</sup> listed as reserved in Chapter 6 are not reserved in the Windows-based COE.

---

<sup>8</sup> The COE is migrating to implement environment variables, defined in Chapter 6 for backward compatibility with legacy segments, as registry entries. The target NT-based COE will use registry entries accessed through the appropriate APIs rather than environment variables. Developers are advised to begin migrating towards this approach.

Segments shall utilize the standard Windows environment variables. Segments shall not create environment variables with the same name as any reserved environment variable.

The Windows operating system uses three levels of environment variables:

1. System environment variables,
2. User environment variables, and
3. AUTOEXEC.BAT/.NT environment variables

System environment variables are set during installation and administrators and developers using the appropriate Win32 API can change their values. The system environment variables listed below are reserved keywords for Windows COE platforms:

- COMPUTERNAME
- ComSpec
- NUMBER\_OF\_PROCESSORS
- OS
- Os2LibPath
- PROCESSOR\_ARCHITECTURE
- PROCESSOR\_IDENTIFIER
- PROCESSOR\_LEVEL
- PROCESSOR\_REVISION
- ProgramFiles
- TEMP
- windir.

Segments may not change the above system environment variables.

User environment variables can be set by users and administrators and take precedence over system environment variables. The system and user environment variables listed below are reserved keywords for Windows COE platforms:

- ALLUSERSPROFILE
- APPDATA
- HOMEDRIVE
- HOMEPATH
- HOMESHARE
- LOGONSERVER
- Path
- PATHEXT
- PROMPT
- SystemDrive
- SystemRoot

- USERDOMAIN
- USERNAME
- USERPROFILE.

These environment variables may be changed or extended in accordance with standard Windows practices.

The root-level AUTOEXEC.BAT and CONFIG.SYS files, used during boot up to create the Windows environment, are reserved files and shall not be modified by any segment, excepting COTS segments. AUTOEXEC.NT and CONFIG.NT files set the user environment for the command prompt and are reserved files and shall not be modified by any segment. Segments, excepting COTS segments, should not set or extend the Windows path environment variable. Dynamic link libraries (DLLs) and other components must put information in the Windows registry that will tell the operating system where the DLL or EXE code for the object is located, and how it is to be launched.

All Windows system INI files (specifically, WIN.INI and SYSTEM.INI) are reserved files and shall not be modified by any segment, excepting COTS segments. Segments may create and modify their own local INI files for segment variables not suited for the registry in accordance with the Windows Logo Program.

### 7.5.3 Reserved Filenames

Windows core files are reserved filenames and shall not be used as the name of a segment file. Windows core files are files required by the Windows kernel and infrastructure to support segments and system operations. Core files are installed during the OS installation and are provided during OS upgrades; e.g., DirectX and Direct3D. The list of core files is dynamic, and as such a current list of core files is maintained by Microsoft and can be downloaded from the Microsoft home page.

In addition to the core files, the Windows OS's reserves the following filenames:

- CON
- AUX
- COM1
- COM2
- COM3
- COM4
- LPT1
- LPT2
- LPT3
- PRN
- NUL.

These names are file handles used by the OS to access hardware resources available on most PCs. Using these as segment filenames would cause conflicts with printing, serial

port communications, and keyboard input. An attempt to use one of the device names listed above results in a system error message.

#### 7.5.4 Reserved Directory Names

Segments are assigned a directory name when they are registered so as to avoid two developers using the same assigned directory. A current list of assigned directories is available from the COE Engineering Office.

The system root directory is reserved, and segments shall not create new files in the root directory. Additionally, application DLLs may not be installed under the system root.

#### 7.6 Naming Conventions

Every device or user attached to a COE network shall have a unique identification. A duplicate identification for a domain, device, or user can cause network-wide anomalies and result in service interruptions of varying levels of severity. The naming conventions given in this subsection shall apply to all DOD organizations that have established, or plan to establish, network domains using the Windows NT or Windows 2000 operating system (OS) families for COE-based systems.

- It is recommended that all *existing* Windows OS based domains, servers, and workstations be named in accordance with these conventions. Full implementation of the conventions on existing systems is expected to take some time to achieve full DII compliance as platforms are replaced.
- All *planned* or *future* Windows OS domains, servers, and workstations shall comply with these conventions before implementation.

The naming conventions provided in this section are predicated upon the following facts:

1. There may be no DNS or Windows name resolution available.
2. Joint or unified networks shall be supported. These networks may consist of devices from all services and agencies that may originate from different locations.
3. Mobile laptops should be supported. Users should be able to connect and disconnect from COE networks without causing a name conflict.
4. The Windows Logo Program does not address the subject of naming conventions.

Naming conventions for classified and unclassified COE platforms and devices are the same except as noted in the subsections below. COE platforms and devices that may be accessed by non-DOD users (e.g., proxy servers on the Internet) are exempt from all naming conventions as provided in this section.

### 7.6.1 Directory Names

All segments, except COTS, are required to obtain an assigned directory name at segment registration time. The assigned directory, as explained in Chapter 6, is not the same as the segment prefix because segment prefixes are not guaranteed to be unique for all segments.

Segments are to be self-contained underneath the assigned directory. For products in which abbreviated segmentation is done, the “pseudo-segment” that contains the segment descriptor information shall also be given an assigned directory name. Segments should limit the assigned directory name length to less than 32 characters. COTS segments should use the default directory name(s) provided by the vendor.

### 7.6.2 Filenames

Windows provides support for filenames up to 255 characters long (including the path and extension). Segments should limit filename length to less than 32 characters for files created by a segment’s user.

- Filenames can include any character except the following:

\ / : \* ? < > | “

### 7.6.3 Registry Keys

Windows provides support for registry keys and subkeys up to 256 characters long. In the context of this subsection registry keys are classified as:

- Root key - registry key located at the root level of a hive
- Subkey - registry key located under a root key, can be nested under other subkeys
- Top level key - root key or first subkey inserted into the registry by a segment
- Key - registry key, can be a root or subkey

Many registry root keys, subkeys and values must follow the naming conventions dictated by the Windows registry itself. An example is the registration of a file extension. The extension itself is registered as a root key in the HKEY\_CLASSES\_ROOT hive and must be in the format of a period followed by three alphanumeric characters. Top level keys created by a segment shall follow one of the two formats: *SegPrefix-Title* or *SegPrefix* where *SegPrefix* is the segment’s prefix assigned when the segment was registered, where - is the minus sign used as a separator, and *Title* is a descriptive title up to 249 characters in length that is assigned by the segment developer. The segment prefix may be used without a minus sign or title as a top-level Windows registry key. Subsequent subkeys under the top-level key are not required to start with the segment prefix.

**This page is intentionally blank.**