IBM Informix Dynamic Server (IDS), Version 11.50
April 22, 2009

Administering Informix Dynamic Server on Microsoft Windows

Dr. Elisabeth Bach, Advanced Support Engineer
(elisabeth.bach@de.ibm.com)

Guy Bowerman, Informix Architect
(guyb@us.ibm.com)

Jasna Krmpotic, Informix Technical Editor
(jasna@us.ibm.com)

Note: Before using this information and the product it supports, read the information in "Notices".

Table of Contents

Introduction

This document is a resource for administering IBM® Informix® Dynamic Server (IDS) on the Microsoft® Windows® platform. It provides information for DBAs and engineers who are familiar with IDS on Linux® and UNIX® platforms, and for Windows DBAs who require more information about the Windows implementation of IDS than provided by the existing documentation. It is designed to supplement rather than replace product manuals such as the *IBM Informix Migration Guide* and the *IBM Informix Dynamic Server Installation Guide for Windows*.

The Windows platforms referred to in this document are assumed to be 32-bit versions unless otherwise stated. Benefits of newer versions of Windows operating systems (such as Vista and Server 2008) and differences for 64-bit versions will be highlighted where appropriate. The document is current for IDS versions 10 and 11.

Most of the internal workings of IDS are the same on all platforms, as is the use of the onconfig file for the configuration of an instance and the online log for reporting status information and events about the instance.

Chapter 1describes IDS implementation differences between Windows and UNIX platforms. It assumes that you are familiar with the concepts of the Windows operating system. Subsequent chapters deal with installation, security, and performance. A chapter looking at usability, and explaining Windows-specific methods from using the system command in stored procedures to RSS follows. The last chapter explains Windows-specific troubleshooting.

This document is a work in progress. We welcome your feedback and suggestions for topics. Please send comments to guyb@us.ibm.com and elisabeth.bach@de.ibm.com.

1. IDS Implementation for Windows Operating Systems

Generally the internal workings of IDS are the same whether IDS runs on Windows or UNIX. This chapter will look at the key differences, where IDS features have been designed to integrate more closely with the Windows operating system.

The Windows port of IDS has some specific adaptations to the Windows operating system, for example the implementation of an IDS instance as a service, or the use of the Windows registry to store certain configuration information. The installation process is naturally platform specific, as are some of the details of user authentication. On Windows IDS consists of a single operating system process - virtual processors (VPs) are implemented as operating system threads as opposed to UNIX, where VPs are implemented as separate processes.

1.1 Windows Services

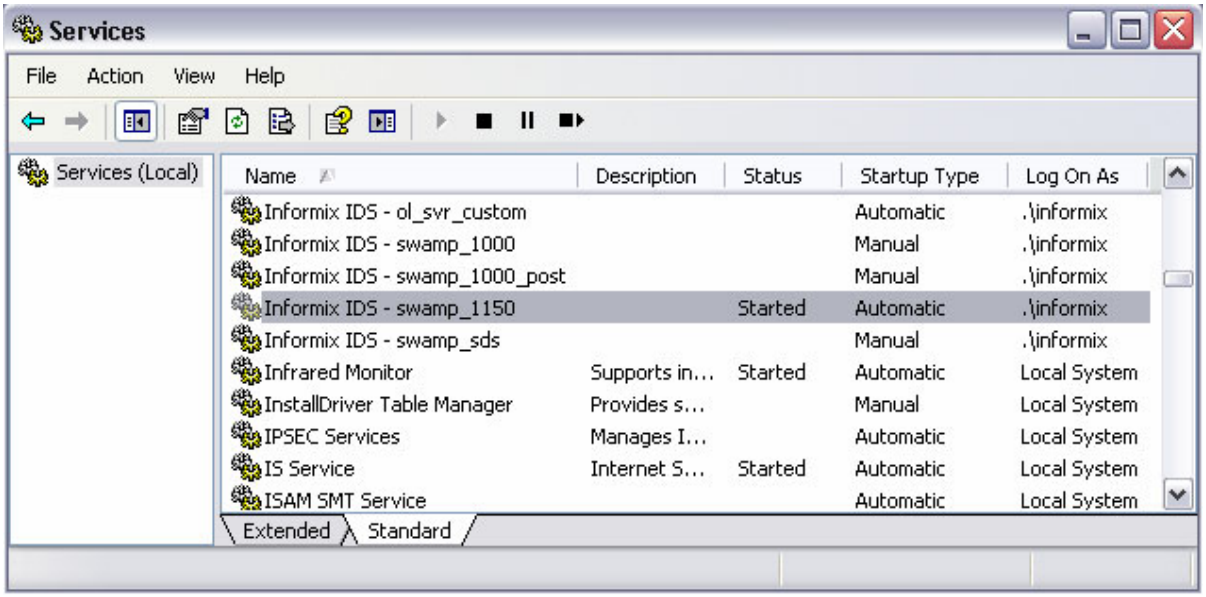

Figure 1 The Windows Services applet

IDS instances on Windows are implemented as operating system services. Windows services are background tasks analogous to UNIX daemons. This mechanism provides the option of automatic startup at boot time without requiring an authenticated user logging into the system and without any additional scripting. The configuration information for services is stored in the Windows Registry, and services can be configured with the Services applet (services.msc).

Services on Windows are implemented with service application programs. The service application programs communicate with the Windows Service Manager. They are not designed to be started on their own. In the subsequent sections, the names of those service application programs will be given for each service implemented by IDS.

When a new IDS instance is created, a corresponding Windows service is defined. The IDS Instance Manager utility creates new instances and takes care of the implementation details. Details about this program are discussed in the IDS *Administration Guide* and later in this document.

A service can be started in several ways. Two ways are provided by the operating system. The Services applet (**services.msc**) is available in the Control Panel. Start parameters can be specified in the service properties window. On the command line, the Windows **net start** command can be used. This operating system command lacks the

option to pass command-line parameters. The Informix installation provides another command-line tool, the **starts.exe** command, which is in %INFORMIXDIR%\bin. This command can parse command-line parameters.

These ways to start a service are not confined to IDS; any service can be started using these methods.

A typical IDS installation on Windows platforms will create the following services:

- Informix IDS service (INFORMIXSERVER)
- IDS Messaging service
- ISM Local Execution service
- ISM Portmapper service
- ISM Server service
- IBM Informix Server Discovery Service for SNMP

1.1.1 Informix IDS - %INFORMIXSERVER%

This is the main Windows service for IDS, which launches the oninit process. There is one IDS service for each IDS instance installed on the computer and the service name always matches the value of the INFORMIXSERVER environment variable. Creating a new instance with the instance manager will add a new service. Starting the IDS service causes the **oninit.exe** process to be launched as a background process. Stopping the service causes the command **onmode –kuy** to be executed to shut down IDS. This service can be set to log on as the informix user or as the LocalSystem account.

The service application program for IDS is called **onscpah.exe**. When it starts, **onscpah.exe** launches the IDS **oninit.exe** process, so for every instance started as a service on Windows there will be two processes running: **onscpah.exe** and **oninit.exe**. Details about the implementation can be found in Chapter 1.8.

As mentioned above, there are different ways to start the Informix Service:

- Starting with IDS 11.50.xC4, the **oninit -w** command has the same usage as it does on UNIX. It will start the IDS service and wait in the foreground until IDS is initialized. It will then return to the command prompt. This feature is useful for embedded systems that need to launch IDS and wait until initialization is complete without user interaction.

- Using the Services applet (**services.msc**) in the Control Panel. The oninit start parameters such as **–iy** and **-s** can be specified in the service properties. In IDS 11, this service is set to start automatically when the computer is booted if the “Initialize Server” option is chosen during installation. Otherwise, the service will be set to start manually.

  IDS can be set to start manually or automatically when the computer is booted by adjusting the service properties in the Services applet.

- Using the **starts.exe** command in %INFORMIXDIR%\bin. This command takes the IDS instance name and any oninit start parameters as arguments. For example, use the following command to start the current instance in single user mode: starts %INFORMIXSERVER% -s

- With the Windows **net start** command. This operating system command can be used to start any Windows service, but it lacks the option to pass command-line parameters, so is generally less useful than the **starts.exe** command.

Note that IDS can also be started in the foreground by running **oninit.exe** from a command window. This way of running IDS independently of the Service Control Manager is only for diagnostic purposes.

In addition to the instance-specific services the following services are created by the installation:

- IDS Messaging service

- ISM Local Execution service
- ISM Portmapper service
- ISM Server service
- IBM Informix Server Discovery Service for SNMP

1.1.2 IDS Messaging Service

The Messaging service (**msgserv.exe**) sends IDS status messages to the Windows Application Event Log. The same messages will appear in the event log and the Online log file determined by the MSGPATH onconfig parameter. If you do not require IDS status messages to be recorded in the Application Event Log, this service can be disabled. Messages will continue to be written to the Online log. The messaging service is set to start manually, and will be started by the oninit process when an instance is brought online.

1.1.3 ISM Local Execution Service

The ISM Local Execution Service (**nsrexecd.exe)** is an Informix Storage Manager (ISM) component that receives ISM commands to execute with the network. This service can be disabled if you do not use Onbar for archiving, or you use Onbar with another storage manager such as Tivoli®.

1.1.4 ISM Portmapper Service

The Portmapper service (**portmap.exe**) is used by ISM to convert Remote Procedure Calls (RPC) into TCP Port numbers. A default IDS instance sets this service to start automatically.

If this service fails to start, check if there is any UNIX emulation software installed, such as Cygwin or Microsoft Services for UNIX, which also implement portmapping and would cause a TCP port clash.

If you do not use OnBar with ISM, the Portmapper service can be disabled.

1.1.5 ISM Server Service

The ISM Server (**nsrd.exe**) is an ISM component that emulates an RPC-based save and recover service. This service can be disabled if OnBar with ISM are not used on the computer.

1.1.6 IBM Informix Server Discovery Service for SNMP

This service (**onsrvapd.exe**) is installed only if the Windows SNMP service is running when you install IDS. It is used to facilitate remote monitoring through SNMP and can be disabled or deleted if you do not use SNMP to monitor IDS.

1.2 Windows Registry

The registry is a centralized place for storing configuration information on the Windows platform. It is organized into several hives. The most important ones are HKEY_LOCAL_MACHINE (known as HKLM) for computer-specific settings, and HKEY_CURRENT_USER (also known as HKCU) for user-specific settings.

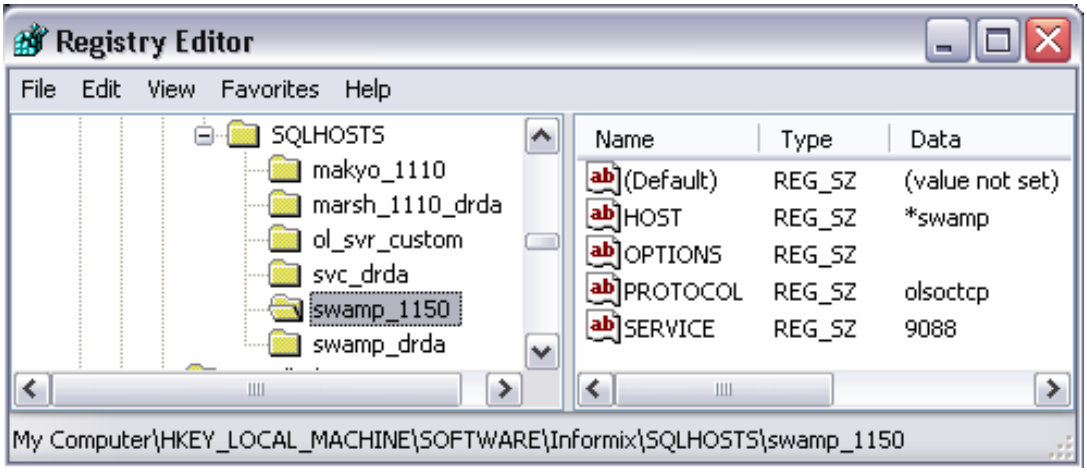

Figure 2 Registry key for an IDS SQLHOSTS entry

On Windows platforms, IDS uses the registry to store some of its configuration information, for example information about installation directories, sqlhosts settings, and service definitions. The registry keys used by IDS are created when IDS is installed and do not need to be directly edited during regular database administration activities.

If you need to edit the registry for any reason, the default GUI editing utility is **regedit.exe**. It includes the ability to export and import registry keys. Various command-line utilities are available to script registry modifications. One example is **reg.exe,** which is included with Windows XP and Server 2003.

The following sections describe in more detail how IDS uses the registry.

1.2.1 Informix Key - HKLM\Software\Informix

Most system-wide IDS and Client SDK or Informix Connect (I-Connect) registry configuration information is placed under **HKLM\Software\Informix** at install time. The main sub-keys are:

- **Online** – (IDS)
- **SQLHOSTS** – (IDS and Clients)
- **DBMS** – (IDS)
- **Setup Framework** (IDS)
- **Environment** - (Clients)
- **netrc** – (Clients)

Other sub-keys, such as BladePack, ESQL/C, Informix ORDBMS, Informix-Connect, and Setup, are little more than placeholders for products such as the Client SDK to record their versions and installation directories.

1.2.2 IDS Instance Key – Informix\Online\%INFORMIXSERVER%

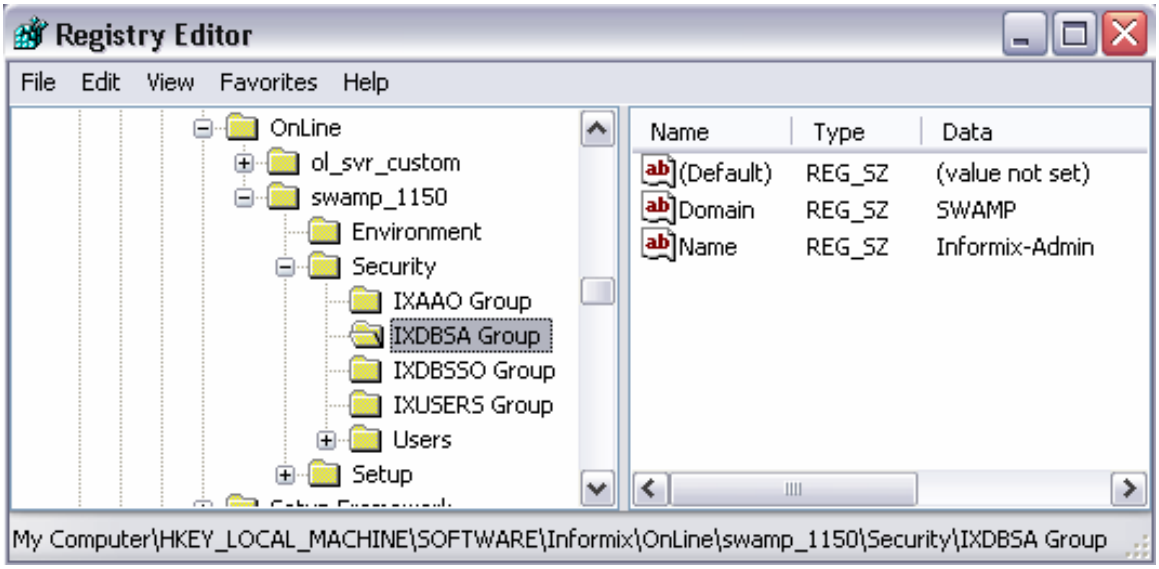

Figure 3 An IDS instance registry sub-key showing the IXDBSA Group

Every IDS instance created on a computer will have its own registry key under HKLM\Software\Informix\Online. The key name must match the value of the **INFORMIXSERVER environment variable**. If this key is not present, or if the environment variable is not set or is set incorrectly, you will see the following error when you run an Informix utility such as **onstat**:

ERROR: Could not initialize the security subsystem. Please ensure that this account has the necessary privileges and ensure INFORMIXSERVER value exists in the registry and environment.

The INFORMIXSERVER key contains the following sub-keys:

- **Environment**
  The values under this key are environment variables that are read by the IDS service before starting the **oninit.exe** process. As on UNIX platforms, values such as **INFORMIXDIR** and **ONCONFIG** are used to locate and read the onconfig file. Values such as **DB_LOCALE** determine the database locale, which must match the database locale set at the client. These are independent of the language of the installed Windows version. For example, the **CLIENT_LOCALE** environment variable is set to EN_US.CP1252 by default even when installing on a German or French version of Windows. The following figure shows the default values used. Note that you can add environment values you want to set specifically for this instance here.

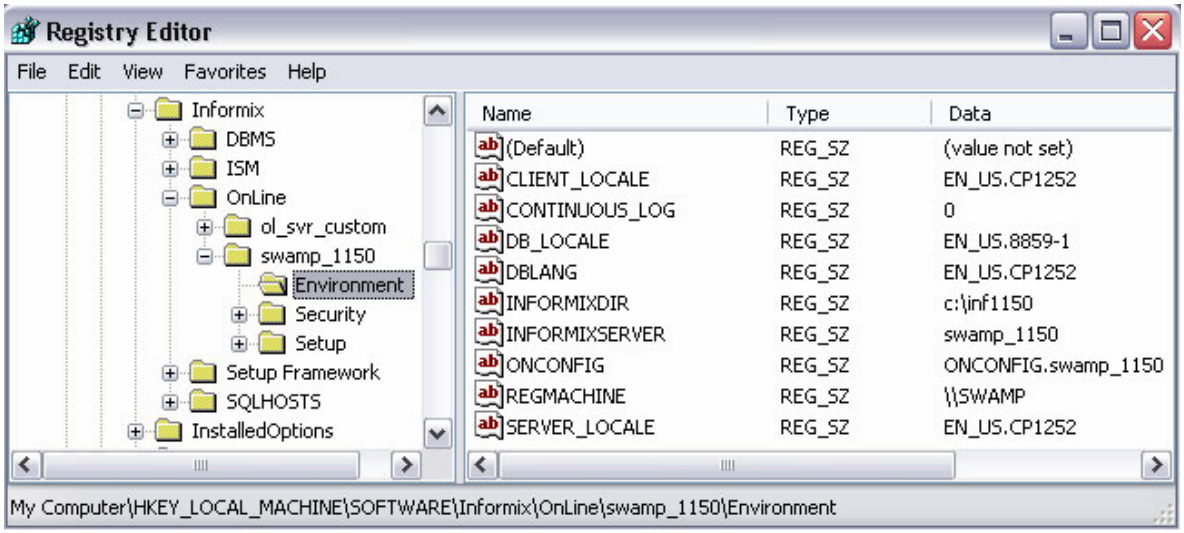

Figure 4 An IDS instance registry sub-key showing the environment settings

- **Security**
  The Security key has a sub-key for each group used by the Role Separation feature**: IXAAO Group, IXDBSA Group, IXDBSSO Group,** and **IXUSERS Group.** Within each sub-key is a value called **Name**, which contains the value of a Windows security group.

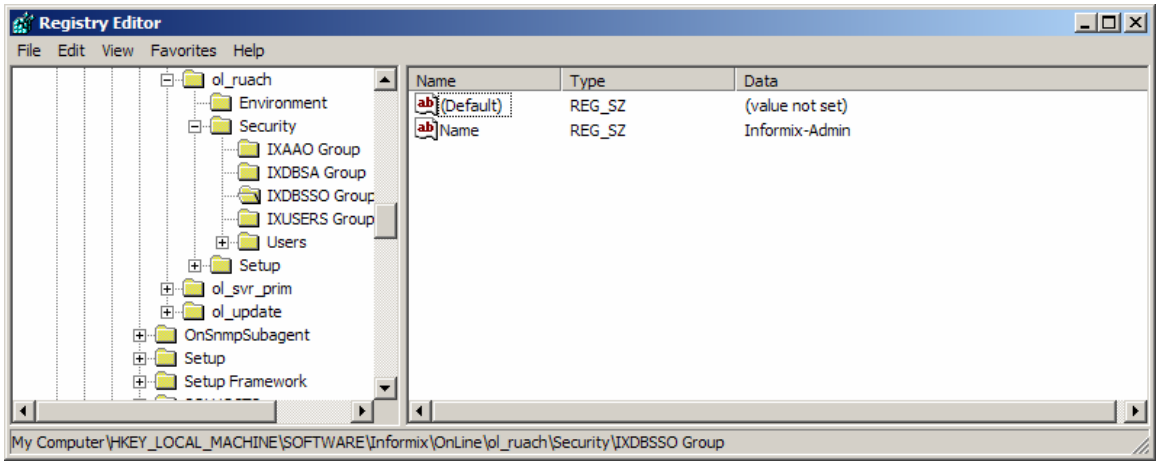

Figure 5 An IDS instance registry sub-key showing the security settings

When Role Separation is not enabled, the Name value in the **IXAAO Group, IXDBSA Group,** and **IXDBSSO Group** is set to the Windows group "Informix-Admin", which is created at install time. This means that members of the Informix-Admin group can perform all of the AAO, DBSA, and DBSSO tasks. If Role Separation is selected at install time, different groups can be created for each of these roles. Refer to the *Trusted Facility Guide* to learn about the functions of these roles, which are generic across all platforms.

One advantage on Windows platforms is that Role Separation can be partially implemented by assigning a different operating system group to the values of some but not all of these keys.

Without Role Separation, the **IXUSERS Group** is assigned to '*' – meaning any valid user can access IDS. If a specific operating system group is set under this registry key, only members of that group can connect to IDS.

The IXDBSA Group key has one additional value: **Domain**. If the local computer is used to validate user access (that is, if IDS was installed on a computer that is not part of a Windows domain, or "Local Installation" was chosen at install time) the *Domain* value is set to the name of the local computer. If user validation is performed at a domain controller level (IDS was installed on a Primary Domain Controller or "Domain installation" was chosen at install time), this value is set to the name of the Windows domain.

Also under the Security key is a **Users** key – this acts as a form of cache to store internal user ID representations of previously connected users to reduce connection time. No login credentials are cached and authentication is not bypassed.

- **Setup**
  Under this key are records of instance-specific file names, services, Program Manager group icons, and registry keys that are used by the uninstall program to cleanly remove IDS and any instances created.

1.2.3 SQLHOSTS Key – Informix\SQLHOSTS

The **SQLHOSTS** registry key stores information about connections for defined servers. On UNIX platforms, this information is found in an sqlhosts file. Clients as well as the IDS server access the same registry key. When IDS is installed, a sub-key is created under **SQLHOSTS** with the same name as the INFORMIXSERVER value. Every sqlhosts sub-key represents a separate server definition.

The recommended way to edit the SQLHOSTS entries is to use the **setnet32** utility, which is supplied with I-Connect and the Informix Client SDK. However, it is sometimes necessary to add or change values manually, such as group entries for Enterprise Replication. In such cases refer to the usage of the registry tools mentioned above.

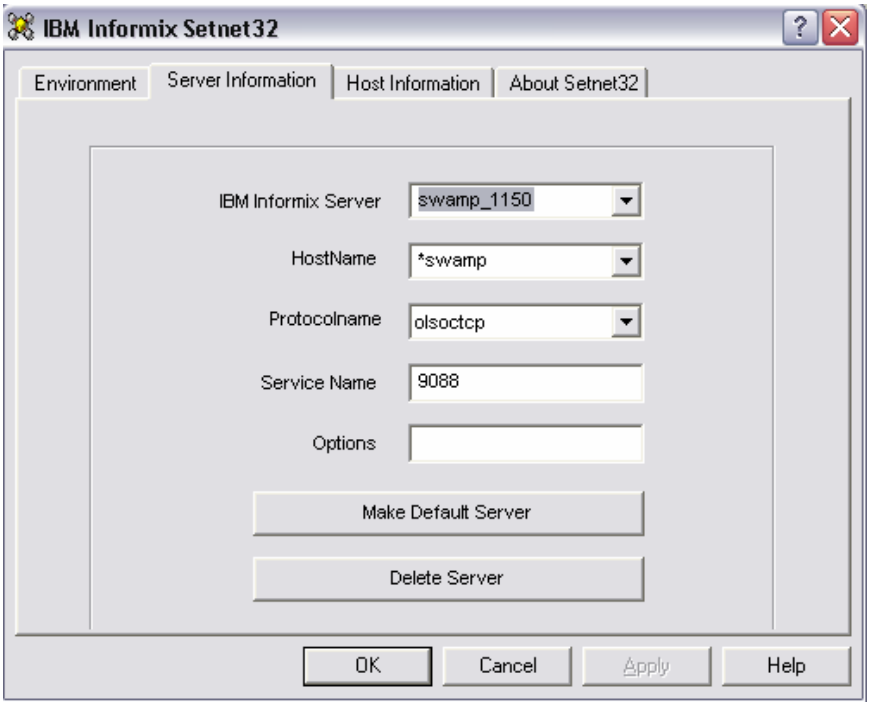

Figure 6 The Setnet32 program Server Information panel

The Server Information panel in setnet32 represents a sqlhosts entry – entering a new value for **IBM Informix Server** in this panel creates a new registry key of the same name under **SQLHOSTS**.

1.2.4 DBMS Key – Informix\DBMS

In IDS versions before version 11, the **DBMS** key stores information to be used by the IDS uninstall program to locate files and other operating system resources to be deleted when uninstalling IDS. Information under this key is also used by the IDS setup program to upgrade an instance.

The DBMS key also contains certain configuration information, such as whether the IDS installation is a Windows Cluster installation, whether a domain or a local installation was performed and version information that is used when reverting to an earlier version of IDS. The Instance Manager program (**instmgr.exe**) uses the DBMS key as a template for computer-wide information of this nature when creating a new instance.

If data under this key is corrupted, for example after a failed IDS installation, it can cause problems for subsequent installations and upgrades. If this should happen, the key should be deleted.

1.2.5 Service Settings

Windows service information is also stored in the registry. The settings for all installed services can be found at **HKLM\System\CurrentControlSet\Services**, the settings for the instances are found looking for the name of the service. These details can also be viewed when looking at the properties settings of an installed service (in the Services applet), and include details about the start type of the service, the image path, and its name.

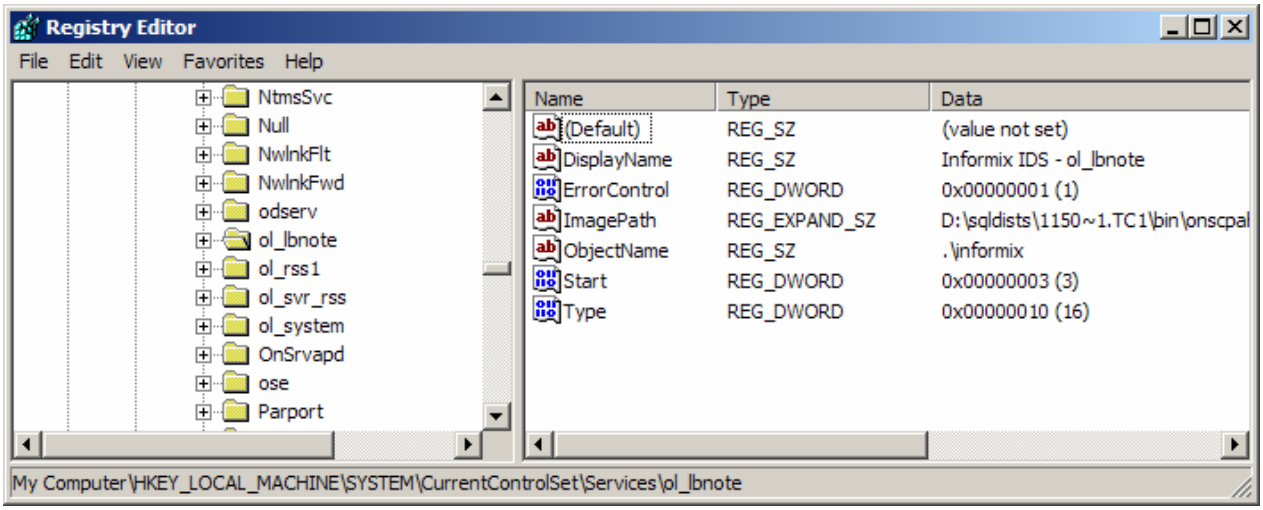

Figure 7 The service entry for an instance called ol_lbnote

1.2.6 Installing 32-bit IDS on 64-bit Windows Operating Systems

To support running 32-bit software on 64-bit Windows operating systems, Windows has a facility called WOW (Windows 32 on Windows 64). It consists of a set of DLLs that are used to hook all the code paths where 32-bit code would transition to the native 64-bit system. These DLLs do the mapping to the 32-bit calls needed to run. This mapping also influences the registry, so that 32-bit registry keys are stored in a different location from the 64-bit keys.

When installing a 32-bit instance on 64-bit Windows, the registry keys normally located in HKLM\software are relocated automatically to \HKLM\wow6432\software. This is a feature of the operating system that makes sure that the software can access the keys when asking for any keys under HKLM\Software. However, services entries are created in \HKLM\CurrentControlSet\Services, just like their 64-bit counterparts.

1.3 Windows File System

The IDS I/O subsystem implementation on Windows platforms is implemented using overlapped or asynchronous operations with no operating system buffering. This could be summarized as using KAIO with cooked files, or equivalent to the UNIX DIRECT_IO setting.

IDS supports both cooked files and raw devices as dbspaces. See section 1.3.2 for more details.

1.3.1 Kernel Asynchronous I/O (KAIO)

The database server performs I/O to dbspaces asynchronously, meaning that I/O is queued and performed independently of the thread that requests the I/O. For performance reasons, KAIO on Windows uses I/O Completion Ports for notification. With this mechanism, a file is associated with a completion port and the asynchronous I/O request for this port are processed by a pool of kernel threads

By default, I/O operations are cached using a system file cache. For dbspace I/O it is preferable not to use this mechanism, as IDS does its own buffering. However, this can mean operations that do not benefit from IDS buffering, such as the use of temp dbspaces to perform sorts, can actually be slower than using the file system. This is worth considering when configuring temporary dbspaces for an application with performance-sensitive sorts. For better performance, consider setting PSORT_DBTEMP to use the file system.

As with other platforms, non-chunk writes, such as writes to log files, are performed using the AIO VP. The number of AIO VPs on Windows is set to 1 by default, but on systems with intensive logging, performance can sometimes be improved by increasing the number of AIO VPs. Note that there is a hardcoded maximum limit of 16 AIO VPs on Windows platforms to avoid accidental over-configuration.

1.3.2 NTFS vs. FAT32 vs. Raw Device

IDS should be installed on an NTFS partition. The FAT32 file system does not support the level of file security required by IDS for product files; however it can be used, and one way to install to a FAT32 file system is to use the IDS Lightweight Installer (see section 2.1.3).

IDS chunk files must be created either on an NTFS formatted file system or on a raw device. On Windows platforms, the performance difference in using a file system versus a raw device for storing data is very small. For this reason, use NTFS to store data for ease of administration.

By default, IDS chunk files are stored in a directory called \ifmxdata\%INFORMIXSERVER%.

The drive is chosen at install time, as is an optional mirror drive. This location is set by the IDS setup utilities **install** and **instmgr**, but dbspaces and chunks can be created in any location specified by using the **onspaces** utility.

1.3.3 Page Size

On Windows platforms, the default IDS page size is 4 KB. On Linux and some other UNIX platforms the default page size is 2 KB. If an application migrates from a 2 KB platform to a 4 KB platform, certain configuration parameters will need to be tuned. Examples are BUFFERPOOL and read ahead variables (RA_PAGES and RA_THRESHOLD).

1.3.4 Avoiding Fragmentation of Chunk Files

When creating a new chunk or dbspace, make sure there is sufficient contiguous space in the NTFS file system it is being created in. If a chunk is created on a disk with insufficient contiguous space, the file could be fragmented, resulting in reduced performance. To analyze or defragment a disk, you can use the Windows Disk Defragmenter, which in XP can be found in the Computer Management Console. On other Windows versions, check the Windows help. Sysinternals also has a tool called **contig**, which can be used to defragment single files. To avoid access problems, work on chunk files only when the instance is down.

1.3.5 Disk Striping vs. Data Fragmentation

IDS supports the dynamic disk striping and mirroring functionality available on Windows platforms, and makes no specific recommendation on whether to make use of these features or rely on data fragmentation to distribute I/O between physical devices to optimize performance. Either option requires appropriate configuration.

1.4 Windows 32-bit Memory Architecture

IDS uses shared memory segments for its internal memory. The Windows operating system implements shared memory as memory mapped files. This is done using a structure called section object. Therefore, when you look at the details of a process you would find that the IDS segments are called sections.

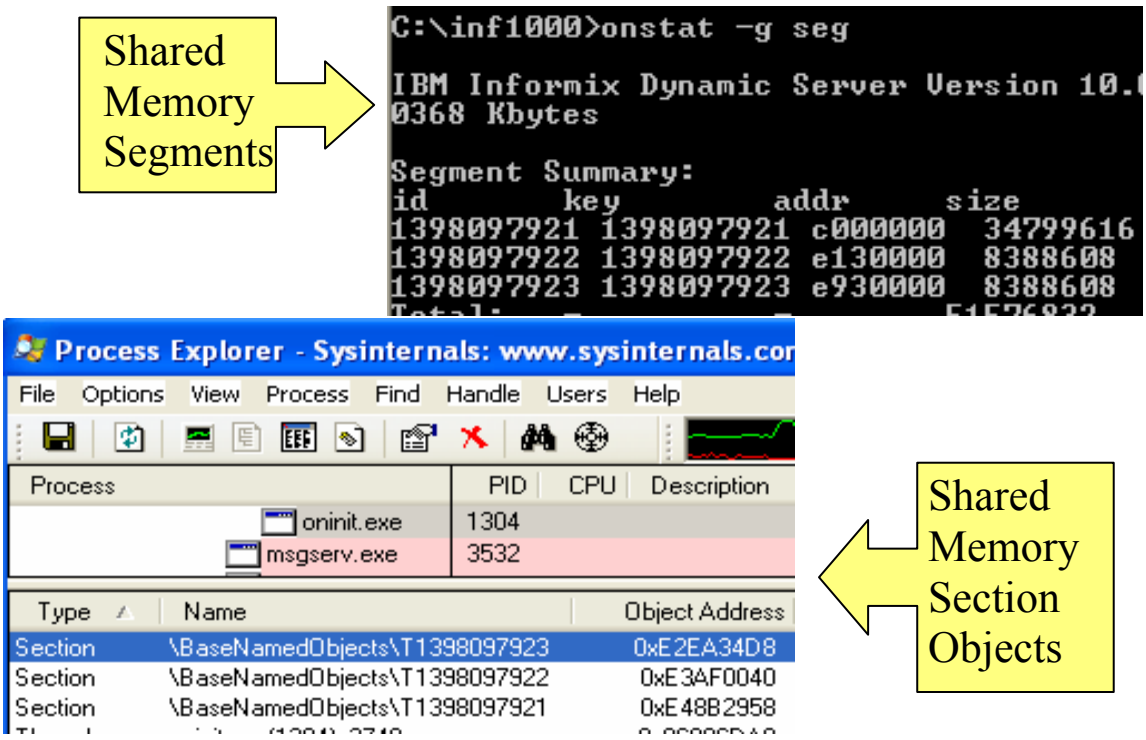

Figure 8 IDS Shared memory segments and Section Objects

In the above picture, the shared-memory segment output from **onstat -g seg** is contrasted with an operating system view of the same shared memory segments shown by the Windows Sysinternals Process Explorer utility. When the section object is mapped the name of the each file mapping object corresponds to the unique shared memory key (which is created from the IDS instance number SERVERNUM).

1.4.1 Windows 32-bit Address Space

On 32-bit versions of Windows operating systems, processes have a theoretical address space of 4 GB. By default, a process is allowed to address 2 GB of this space for its own purposes and the remaining 2 GB are reserved for system use. Typically, the top 256 MB of user space (above 0x70000000) are reserved for operating system DLLs and 64 KB at the beginning are also reserved.

0x00000000 -> 0x000001000 Reserved
0x00001000 -> 0x6FFFFFF User Space
0x70000000 -> 0x7FFFFFF System DLLs
0x80000000 -> 0xFFFFFFF System space

If Windows is booted with the /3GB boot.ini switch, an additional 1 GB is available to user processes, so the addressable space has the following layout:

0x00000000 -> 0x000001000 Reserved
0x00001000 -> 0x6FFFFFF User Space
0x70000000 -> 0x7FFFFFF System DLLs
0x80000000 -> 0xBFFFFFF User Space
0xC0000000 -> 0xFFFFFFF System space

1.4.2 IDS Shared Memory

The oninit utility needs memory for a number of objects, for example buffers to cache data pages, locks, caches for tables or information for current user sessions. This memory is split into two segments, the resident and the virtual segment. The size of the resident segment is determined by various onconfig parameters like BUFFERPOOL and LOCKS. In 32-bit environments, the size of the virtual segment is initially set by the SHMVIRTSIZE parameter, while in 64-bit environments, part of the buffers can also be allocated in the virtual segment. The default **SHMBASE** value on Windows operating systems is 0xC000000, which is where oninit puts the first resident segment.

Processes that attach to IDS shared memory segments, such as oninit, onstat, or onbar, require contiguous chunks of free address space of size equal to the segment size. So if onstat connects to the resident segment and *onstat -g seg* shows it to be 1GB in size (due to the number of buffers and locks configured), the onstat address space will need a 1 GB contiguous interval where no DLLs are loaded. The first place it will try and attach the resident segment is the value of the onconfig parameter SHMBASE. If this fails, it will try and attach at an address allocated by the operating system.

The address space of a process is limited by the DLLs it loads, which fragment the contiguous space depending on the address they are loaded at. The load address, or base address, of a DLL is determined at link time. If a process tries to load a DLL at an address that is already being used, it will be dynamically rebased by the operating system and loaded elsewhere.

You can check the load location of DLL with the Sysinternals tools **listdlls** and **process explorer**. You can also manually rebase DLLs, see the discussion later in the troubleshooting section.

For example, assume that the output of listdlls for a running oninit process shows the system DLLs starting at addresses above 0x7c000000 and **oninit.exe** occupying the lower end of the user space at 0x00400000, with contiguous memory in between. Note that the start address of the **oninit.exe** shows you where this process is located in the memory, and that this start address differs from the start address of the shared memory.

See section 6.4 for a discussion of problems related to DLL locations.

1.5 Windows 64-bit Memory Architecture

On Windows 64-bit operating systems, user processes have 8 TB of addressable space. The maximum shared memory size for a 64-bit Windows port is therefore typically determined by the available physical memory and performance considerations.

As an example, consider the following 64-bit ONCONFIG shared memory settings:

```
SHMBASE 0x80000000L
SHMVIRTSIZE 1044992
SHMADD 261248
BUFFERPOOL size=4K,buffers=1000000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

In those settings, 1 million 4 KB buffers are allocated and the initial virtual shared memory segment size is set to 1 GB. The resultant **onstat –g seg** output becomes:

```
IBM Informix Dynamic Server Version 11.50.FC2     -- On-Line -- Up 00:01:39 -- 5296832 Kbytes

Segment Summary:
id          key         addr              size              ovhd     class blkused  blkfree
1381517313  52584801    80000000          3221225472        38184224 R     783190   3242
1381517314  52584802    141050000         2202730496        25815088 V     286756   251020
Total:      -           -                 5423955968        -        -     1069946  254262
```

The total shared memory size is approximately 5.5 GB, with the segments allocated above the old 2 GB limit of 0x80000000. Note that some of the buffers have been allocated in the virtual segment.

Viewing the DLL load addresses for the **oninit.exe** process in Microsoft Sysinternals Process Explorer shows a mixture of DLLs loaded at 32-bit addresses between 0x60000000 and 0x80000000, and 64-bit addresses above 0x7FF00000000. The space between these values is expected to be contiguous free space available to user processes and shared memory.

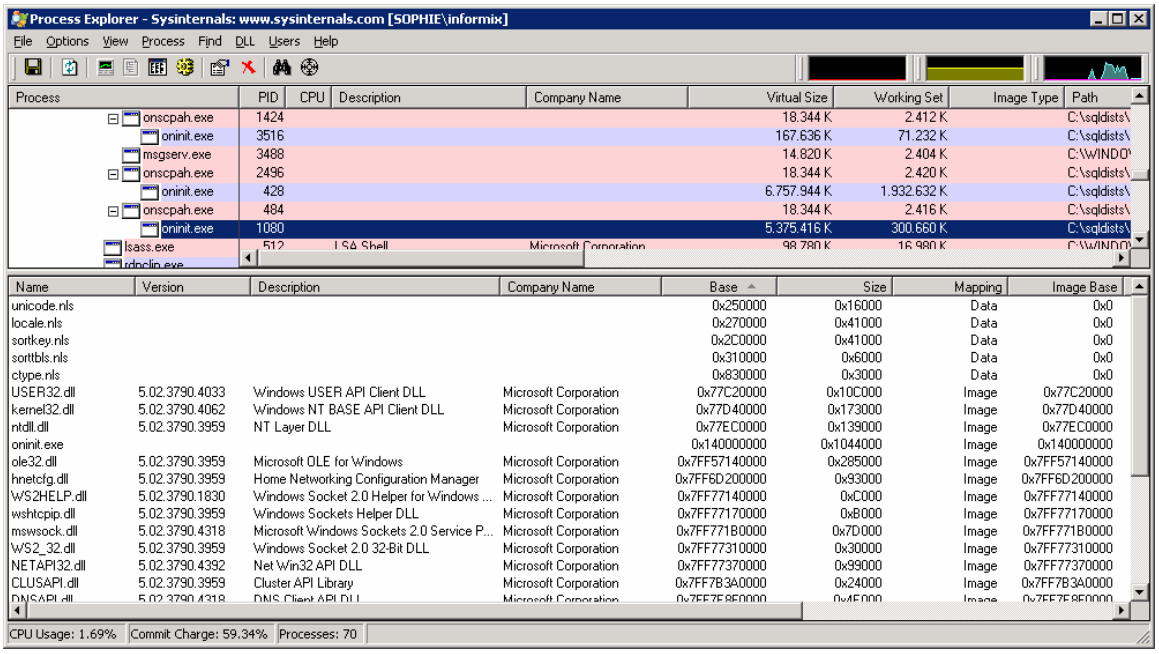

Figure 9 Process Explorer displaying 64-bit DLL load addresses for oninit.exe

1.6 Process Architecture

An IDS instance on Windows has a single oninit process, and each VP is an operating system thread within that process. This is different from UNIX, where each IDS Virtual Processor (VP) runs as an individual operating system process. Therefore, the values in the pids or the IDS VPs displayed using the *onstat –g glo* command represent the thread IDs of operating system threads on Windows.

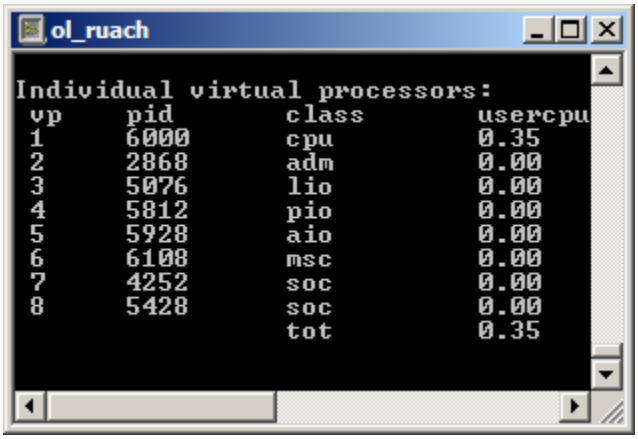

Figure 10 Windows "onstat –g glo" output

The IDS VPs can be viewed as operating system threads by cross-referencing these *"pid"* values with the thread IDs shown by a utility such as Process Explorer:

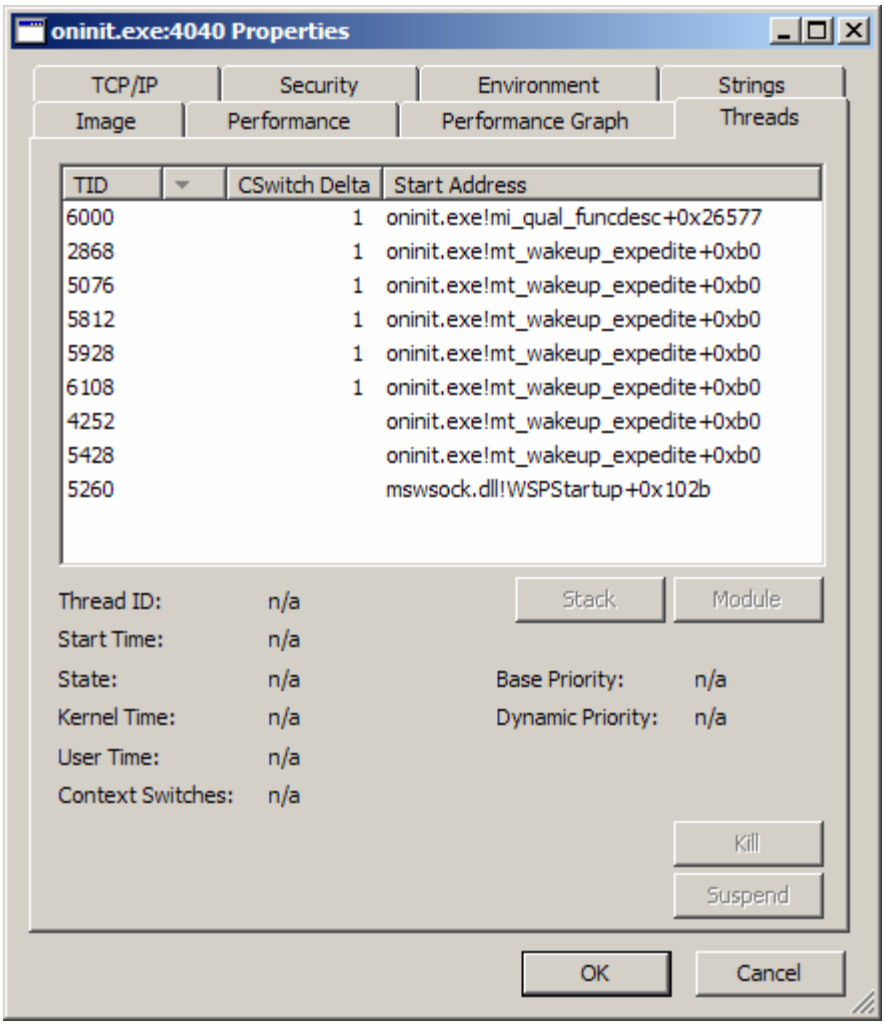

Figure 11 Process Explorer view of oninit.exe threads

In the previous figure, each oninit operating system thread (VP) is shown with its thread ID.
This is the view from the properties window of the Process Explorer. You can also find the oninit operating system threads and their thread ID in the handle view of the Process Explorer. The oninit process sometimes has more than one handle open for an operating system thread, so those threads can be visible more than once in the handle view.

In this section the term “thread” has been prefixed with “operating system” to distinguish it from the Informix “thread”, an execution stream that runs on an Informix Virtual Processor, shown for example by the **onstat –g ath** command.

1.7 Windows Networking

1.7.1 Supported Network Protocols

IDS version 11 and later supports the following network protocols on Windows operating systems:

ONSOCTCP – TCP Sockets
ONIPCNMP – Named Pipes

DRSOCTCP – Distributed Relational Database Architecture™ (DRDA)

The Named Pipes protocol is for local connections only and provides a small performance improvement over sockets. To configure Named Pipes, set the PROTOCOL name to onipcnmp in SQLHOSTS by using **setnet32** or by directly editing it.

The DRDA® protocol is used by IBM Data Server clients (also known as Common Clients). This protocol can be configured by setting the correct PROTOCOL Name in Setnet32 when using C-SDK 3.00.TC3 or later. DRDA cannot be used to communicate with IDS servers prior to version 11.

1.7.2 Socket Virtual Processor Implementation on Windows Operating Systems

Suppose the NETTYPE onconfig parameter is set to the following value:

```
NETTYPE        soctcp,2,,NET
```

On Windows platforms, this onconfig setting results in an **onstat –g ath** output with network threads similar to the following:

tid	tcb	rstcb	prty	status	vp-class	name
8	e4b2bf0	0	2	running	9soc	soctcppoll
9	e4e8d70	0	2	running	10soc	soctcpio
10	e595378	0	2	running	11soc	soctcpio
11	e595950	0	2	sleeping forever	1cpu	soctcplst

As with UNIX, a listener thread is running in the CPU VP. However, only one poll thread is created, instead a *Socket I/O Thread (soctcpio)* is created in its own SOC VP for each "poll thread" configured in the onconfig file.

Socket IO Threads handle receive operations for all connections using I/O Completion Ports to receive completion notifications. These threads perform the bulk of the work of servicing network connections on Windows platforms.

The Kernel Asynchronous I/O on Windows Network I/O subsystems are implemented using I/O Completion Ports, which minimize context switches and maximize parallelization.

1.8 IDS Initialization and Shutdown on Windows Operating Systems

As mentioned earlier, IDS instances on Windows are implemented as services. This section describes how IDS startup and shutdown is implemented on Windows, detailing the connection between the Windows service, the registry and the oninit process.

1.8.1 Initialization

Though the oninit.exe process can be launched in the foreground, the preferred method of starting IDS on Windows is with the IDS service.

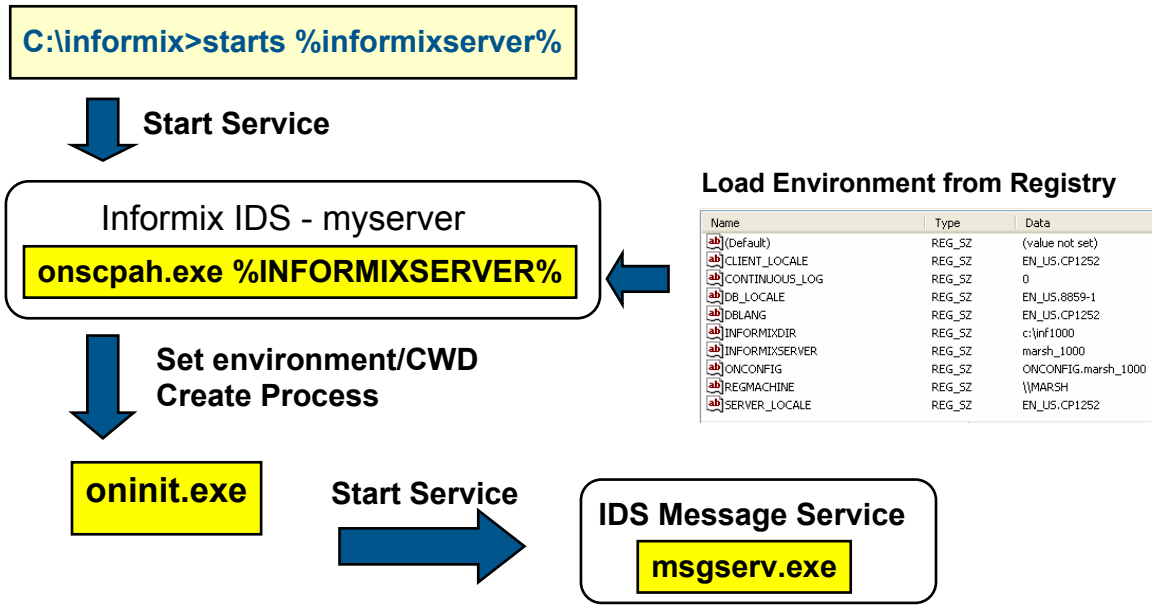

Figure 12 IDS Service initialization sequence

The previous figure shows the sequence of events, beginning with launching the service with the **starts** command. The **starts** command instructs the Windows Service Manager to start the IDS service. The IDS service has an image path pointing to the **onscpah** (Online Service Control Program and Alarm Handler) program – passing the name of the IDS instance (INFORMIXSERVER value) as an argument. This starts the **onscpah** program.

The *onscpah* process read the IDS environment variables from the Environment registry key (see section 1.2.2) and launches the oninit process within this environment. During initialization, the oninit process will instruct the Windows Service Manager to start the IDS Message Service if it is not already started (see section 1.6.2).

The following figure shows the main Windows-specific events during the oninit initialization sequence. This sequence occurs after oninit is launched with the onscpah process running as a service or after oninit is run in the foreground on the command line.

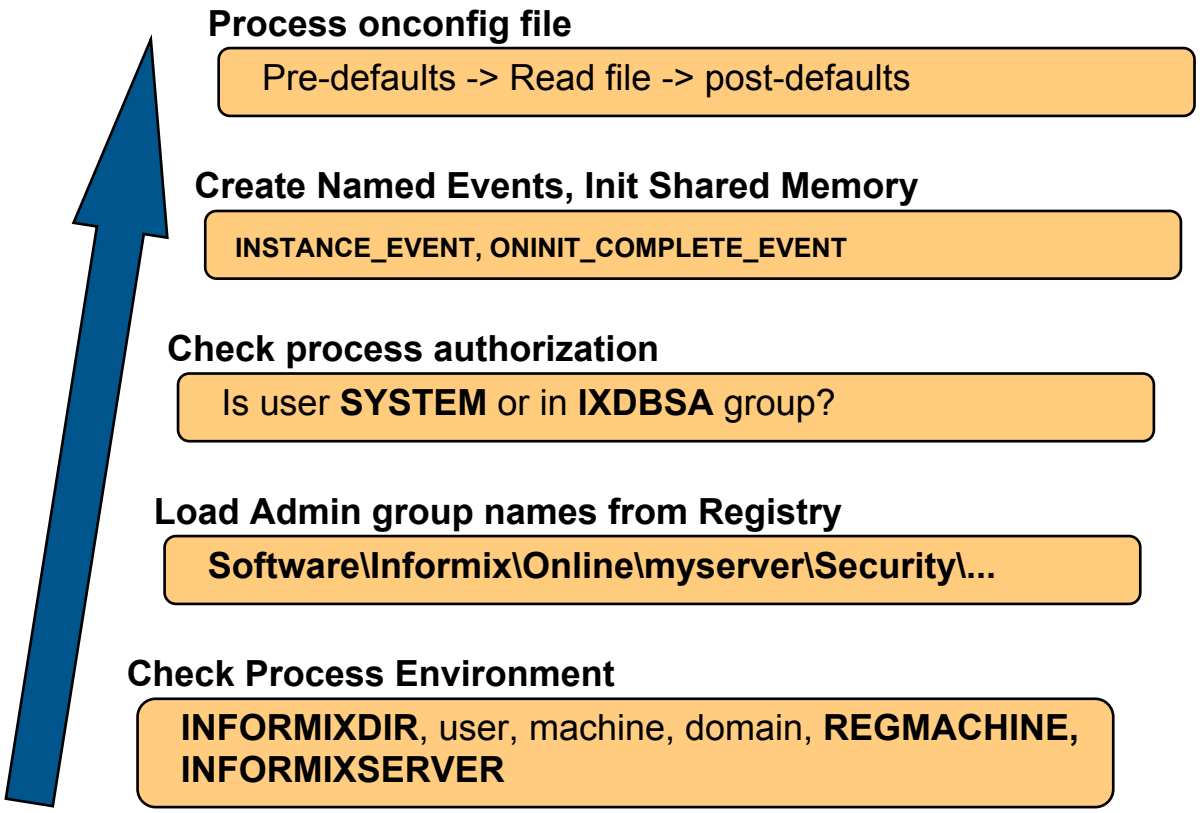

Figure 13 Oninit process initialization sequence

To locate the required registry keys and files, the oninit process checks its environment for the values of INFORMIXDIR, INFORMIXSERVER and REGMACHINE environment variables. It also checks the user credentials of the process owner. The names of the Informix Administration groups (IXDBSA, IXDBSSO, IXAAO – see section 1.1.2) are then loaded from the registry.

At this point, the oninit process checks that the process owner is either the local SYSTEM user or a member of the IXDBSA group (Informix-Admin by default).

Once authorized, the oninit process creates Named Events. Windows Named Events are system-wide synchronization objects used by threads to signal information to waiting threads. IDS sets various Windows events to provide information to external programs such as SNMP to inform them when it is safe to connect to shared memory and when shared memory information should be updated. These Windows Named Events can be viewed in the lower pane of the process explorer.

When shared memory initialization is complete, oninit will begin processing the onconfig file. Before reading the onconfig file, oninit will set default values for parameters such as SHMBASE to ensure that the system will initialize with a minimal configuration. The values in the onconfig file are read using platform independent code, but once read, certain onconfig values will be overwritten to prevent them from exceeding Windows specific limits. Examples include TAPEBLK, which has a minimum size of 8 KB, or NUMAIOVPS, which has a maximum value of 16 (usually only one is required because IDS uses KAIO). If certain required onconfig values are missing at this point, default values are used. An example is the number of CPU VPs (NUMCPUVPS or VPCLASS cpu), which defaults to the physical number of CPUs or CPU cores.

1.8.2 IDS Service Shutdown

The following figure shows the sequence of events at a Windows service/process level when the IDS service is shut down, for example, with the **net stop %INFORMIXSERVER%** command.

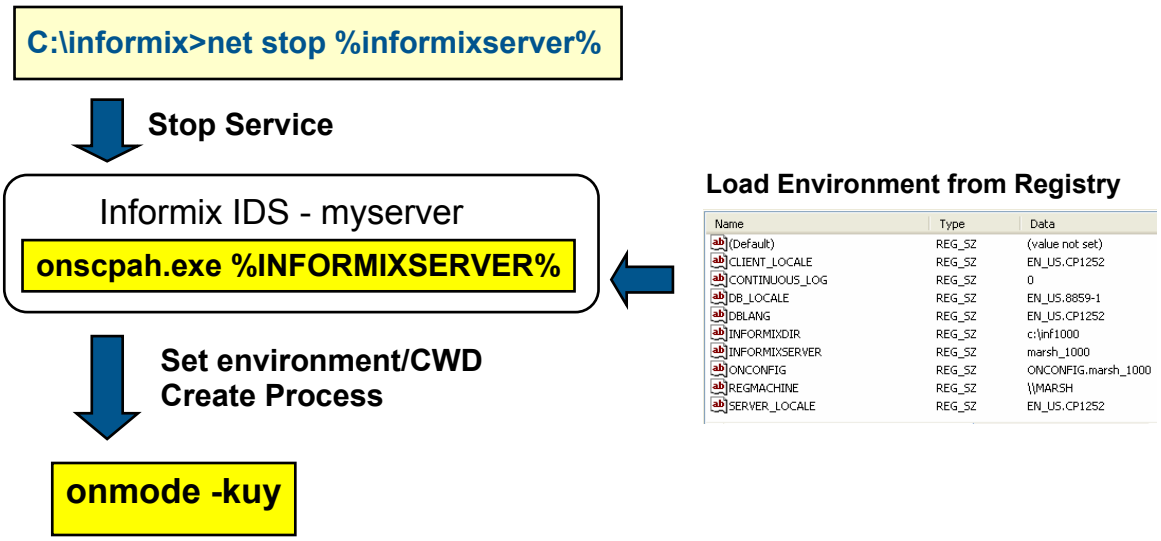

Figure 14 IDS Service shutdown sequence

When the IDS service is shut down, the Windows Service Manager sends a shutdown signal to the *onscpah* process. This causes onscpah to load the same IDS environment from the registry as during startup, and then create a process to issue the **onmode –kuy** command, which will trigger a clean shutdown. When the operating system is shut down, the service control manager informs all service control programs to stop their services. In this case, onscpah will also issue an **onmode –kuy** command to stop the instance.

If the IDS instance is shut down externally (for example, with an **onmode –kuy** command issued from a command window), the IDS service will also shut down.

2. Installation

This section describes how IDS can be installed and customized on Windows operating systems. After introducing the interactive installation options, onconfig parameters that differ from their UNIX counterparts are discussed.

2.1 Installation Options

IDS on Windows operating systems is distributed either as part of a bundle or as a single product. In both cases, the same installer is used for the IDS component.

IDS can be installed either into a domain or as a local installation. These options differ in the location of the user authentication. For a local installation, the security information for the server will reside in the local registry. Also, the informix account will be created locally. If you plan to use role separation, the groups needed for that will also be created locally. However, you do not have to create all users who need to access the IDS instance locally, you can make use of the group system within your active directory and add a global group of users to your local group.

If you have several IDS instances in a Windows domain or require user authentication to be performed centrally, you can install IDS into a domain. This means that the user authentication is done by the Domain controllers. User informix exists only once in the domain, and if you use role separation, the groups are created only once. Also, the SQLHOSTS information can be placed centrally on one computer and can be accessed there.

You cannot change a local installation into a domain installation or vice versa without reinstallation.

Usually the installation creates the user informix, who is part of the administrative group. However, it is also possible to run the services needed by the instance using the LocalSystem account (this is available since version 10.00.TC5). For details how to implement this, see section 3.1.3.

The user informix needs the following advanced privileges and user rights:

- Log in as service
- Act as part of the operating system
- Increase quotas
- Replace a process level token
- Debug programs
- Manage auditing and security log

If there are authentication or startup problems, or it the informix user was created manually, it is worth checking whether these rights are granted. There might be a problem with local policies or some missing rights if an existing user informix was not added during installation. Also, make sure that the password chosen for the informix user during installation adheres to the local password policy, as otherwise the user cannot be created, which will lead to problems during the installation. See also section 3.1.4 and 6.6.

2.1.1 GUI Installer

Informix Dynamic Server is supplied with a GUI installation program **setup.exe**. In IDS version 11 and later the installer is based on InstallShield. Previous versions of IDS use an installer based on an older Microsoft Setup SDK. In both cases the installer has a silent installation option that supports installation of IDS instances with no user interaction. Refer to the product documentation for silent installation usage details.

The Windows based IDS setup program can create a new instance by generating a customized onconfig file from user input during installation (or specified in a silent installation configuration file). If an IDS instance is already present on the computer, the existing instance can optionally be updated.

From IDS version 11 the installation is componentized, enabling users to reduce the overall installation footprint by selecting only those IDS components and languages that are needed, as shown in the following figure.

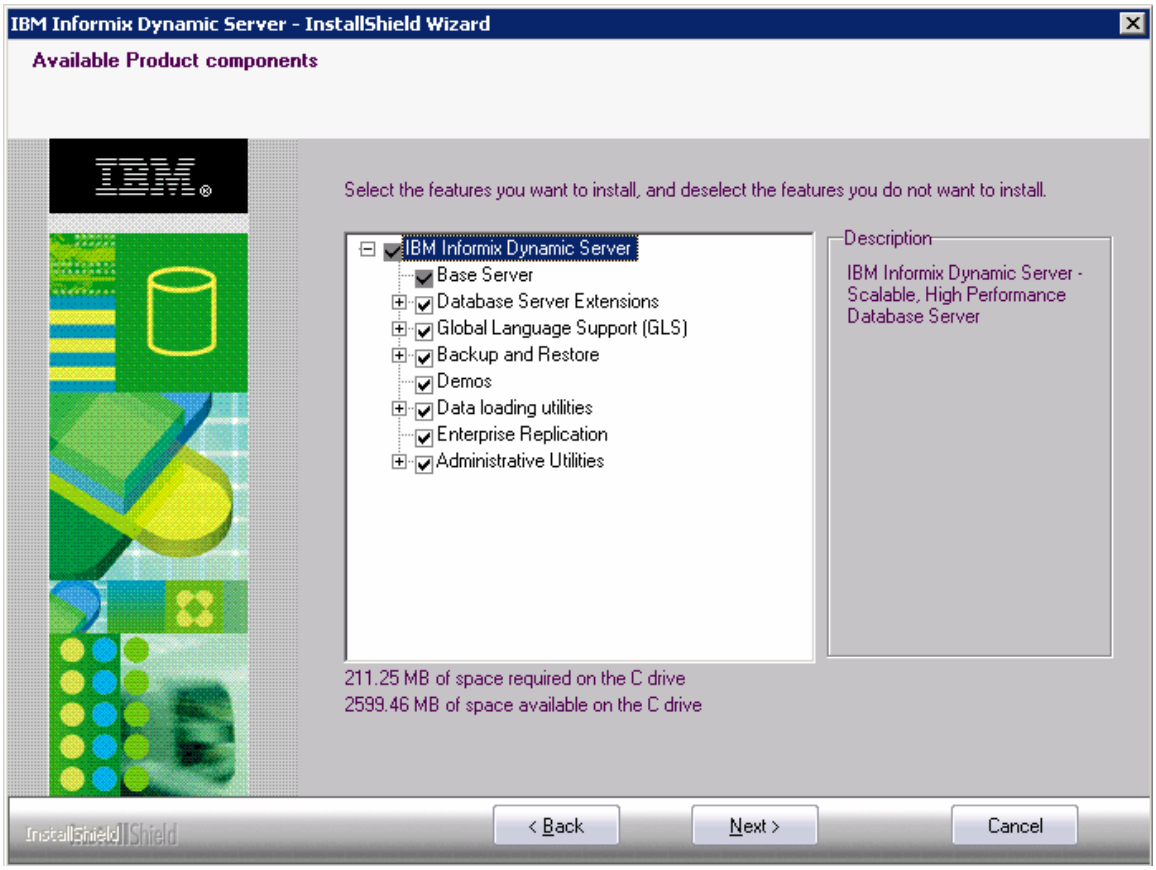

Figure 15 Informix Dynamic Server 11.10 deployment wizard

2.1.2 Multiple Installations of IDS on the Same Windows System

When the IDS installer switched to using InstallShield as its framework in version 11.10, only one copy of each major version of IDS could be installed. You couldn't, for example, have a separate development and production installation using different versions of 11.10 on your computer. You could still create multiple instances with the Instance Manager.

In IDS 11.50.xC1 support for multiple installations was reintroduced with an undocumented **"-*multiple*"** command line argument to the **setup.exe** program. Every time you ran ***setup.exe -multiple*** a new installation would be created in the path of your choice.

From IDS 11.50.xC2 the multiple installation support is once again the default behavior. Any time you run **setup.exe**, if there is an existing installation of IDS on the computer, you will be prompted to create a new instance. If you want to run **setup.exe** to maintain an existing installation, you can use the new *-path* or *-instnum* arguments to identify a specific installation.

Note that for backward compatibility the *"-multiple*" argument is still supported, though no longer required.

2.1.3 Lightweight Installer

In some cases it can be desirable to both configure and initialize an IDS instance prior to deployment, for example when deploying an embedded IDS instance containing pre-configured data to many computers. A free command-line utility known as the Lightweight Installer (**lwt_install.exe**) is available. This utility takes an archive of a pre-configured IDS instance as an argument, and will unzip this file and perform the necessary installation steps (such as creating registry keys, Windows services, users, batch files, and so on) for it to be usable on a target computer.

The lightweight installer makes it possible to transfer completely initialized and configured instances easily from one computer to another. As this is a command line tool, it can easily be used in a batch file. The lightweight installer is available from the IIUG Software repository and can be downloaded, along with a readme file, from http://www.iiug.org/software/index_MISC.html#lwt_install.

2.2 Post-Installation Tasks

During IDS installation, if the option to initialize the server is chosen, a default instance with a data dbspace and a smart blob space is created. With the exception of the onconfig file tuning covered in section 2.3, most of the post-installation tasks relating to the migration of an instance from another platform to Windows such as creating dbspaces and blobspaces, importing data, updating statistics are not platform specific and are covered by the Migration Guide and other documents.

Possible Windows specific tasks connected with running local clients might include:

- Installing the Informix Connect (I-Connect) or the Client Software Development Kit (CSDK) using the Windows GUI installer.

  This will be helpful, as it will give you access to utilities like setnet32.

  If client applications, which have been build without the –static option (available since CSDK 3.00.TC2) run on the server machine , I-Connect or CSDK must be installed. Otherwise the program will fail, as those programs have a run-time dependency on DLLs provide with I-Connect or the CSDK.

- Running Setnet32 to set up client environments.

  Client applications built using ESQL/C look for registry values from the HKEY_CURRENT_USER registry hive to load Informix environment variables used when connecting to IDS. If no values are present they then look in the .DEFAULT registry hive and then the local environment.

  The **setnet32** utility provided with I-Connect and CSDK can be used to define Informix environment values and other settings such as NETRC connection settings in the HKEY_CURRENT_USER Registry hive.

  Another utility called **regcopy** is provided which copies the settings from the HKEY_CURRENT_USER hive to the .DEFAULT hive, allowing all local users in a multi-user to use the same set of "default" environment variables and other connection settings.

- Creating COM objects for VB/ASP/COM applications.

- Configuring the Microsoft Transaction Server (MTS) to work with XA transactions.

  COM objects and MTS assume specific application requirements beyond the scope of this document.

- Defining DSNs with the Control Panel->Administrative Tools->Data Sources applet.

  Unless the client application uses a DSNless connection, ODBC connections for client programs are defined with the Control Panel Data Sources applet.

- Setting system-wide environment variables required by web or other server applications.

  For three tier applications where a web or application server is running on the server machine which connects to a local IDS instance with JDBC, ODBC or PHP/PDO, the application server may need certain environment variables to be set. An example is the PDO (PHP Data Objects) driver which requires INFORMIXDIR to be set in the web server environment. Check the driver documentation to determine specific requirements.

2.3 Onconfig File Differences between Windows and UNIX

Most of the configuration parameters in the onconfig file apply to all platforms. Most differences between an onconfig file on Windows and one on UNIX are syntactical, relating to different PATH structures. This sections looks at the major areas that need to be considered when tuning an onconfig file for Windows.

The Windows install program creates a working onconfig file during installation and when a new instance is created using the Instance Manager (**instmgr.exe**).

2.3.1 File Paths

In the onconfig file, UNIX file paths use forward slashes while Windows file paths use backslashes and drive designators (for example, *C:*). Any parameters that have file paths such as **ROOTPATH**, **MSGPATH,** and **DUMPDIR** are affected.

Note that forward slashes can be used for paths in the onconfig file on Windows if required. However, this might cause problems, for example in the case of error-handling where the ALARMPROGRAM script is triggered. If a path containing forward slashes is passed to a Windows command or to an executable within the Windows command shell, a forward slash can be interpreted as a command argument rather than part of a file path.

2.3.2 4 KB Default Page Size

The default and minimum disk buffer/page size on Windows is 4 KB. If migrating from a platform where the default buffer size is 2 KB special consideration needs to be given to onconfig parameters that are affected by buffer size. Examples include:

BUFFERPOOL – The size=4k BUFFERPOOL setting is the one that sets the number of default buffers on Windows. The number of buffers represents twice as much memory when moving from 2 KB to 4 KB page size.

RA_PAGES / RA_THRESHOLD – The optimum number of pages to read ahead changes when more rows fit on a disk page.

LOCKS – Fewer locks will be required for page-level locking.

2.3.3 Shared Memory

Refer to sections 1.4 and 1.5 for a more detailed discussion of shared memory architecture on Windows and potential problems that need to be addressed when configuring a large shared memory size on 32-bit ports.

The following parameters, which relate to shared memory, require platform-specific consideration:

SHMBASE: The recommended and default setting is 0xC000000 for 32-bit Windows and 0x160000000 for 64-bit Windows

SHMVIRTSIZE: The size of the initial Virtual segment can affect where IDS shared memory is loaded in the 32-bit process address space (see 1.4)

SHMADD/EXTSHMADD: The size of additional shared memory segments may need to be limited if shared memory size is close to the maximum supported 32-bit size

SHMTOTAL: If this value was used on UNIX for a specific reason consider unsetting or re-tuning it for Windows to match your maximum permitted shared memory size

SHMNOACCESS: List of up to 10 memory address ranges that IDS cannot use to attach shared memory. Refer to 6.3.5 for more details

2.3.4 Kernel Asynchronous I/O – KAIO

On Windows operating systems, chunk I/O always uses KAIO; therefore, the **VPCLASS aio** parameter setting will default to 1 if not present. Usually one AIO VP is all that is required on Windows to handle writing to log files, except on systems where large amounts of non-chunk I/O takes place (in which case a small increase can improve performance).

2.3.5 CPU VPs

When moving from a computer with a different number of physical CPUs or cores, re-tune the number of CPU VPs. The following parameters are affected:

VPCLASS cpu – Number of CPU cores. One cpu is a reasonable starting point. Testing is recommended. This value depends on the work performed on the system. Windows supports Processor Affinity. Affinity settings will also need to change if the number of CPUs changes. Note that the noage parameter is not used on Windows.

MULTIPROCESSOR – Relevant if moving from a single-processor computer to a multi-processor computer or vice versa.

2.3.6 Network Virtual Processors

The following points need to be considered when setting the **NETTYPE** onconfig parameter on Windows:

SOCTCP and IPCNMP (Named Pipe) protocols are supported in IDS on Windows. Versions 11.10 and later also support these protocols with DRDA. The UNIX network connection protocols IPCSHM, TLITCP and IPCSTR are not supported on Windows. See also section 1.7.1.

Poll threads only run on the NET VP class. A VP class setting of CPU in the NETTYPE parameter will be ignored.

2.3.7 Tape Devices

Windows tape devices are defined using the Unified Naming Convention (UNC). The onconfig parameters **TAPEDEV** and **LTAPEDEV** are typically set to values such as:

\\.\TAPE0 # The first tape drive on the local machine
NUL # The null device (equivalent to UNIX /dev/null)

Setting the tape parameters to remote devices is not supported.

Both TAPEDEV and LTAPEDEV can be set to write to the file system. In versions prior to 11.10 they write to a file, in version 11.10 and later they can also point to a directory, where the backup will be stored. Refer to the Administration Guide for further information.

2.3.8 J/foundation Libraries

If Java™, stored procedures (known as the J/foundation or Krakatoa feature) are used. It is important that the following parameters are set to the recommended values in the onconfig.std file for your platform. If these values are changed the Java Virtual Machine will fail to initialize:

JVPJAVALIB
JVPJAVAVM

2.3.9 OnBar Library

On Windows operating systems the OnBar utility (if used) loads an XBSA DLL as opposed to a shared object file, so a typical library path setting will look like this**:**

BAR_BSALIB_PATH **C:\ISM\2.20\bin\libbsa.dll** #Location of ISM XBSA dll

If you are on a 64-bit platform, you need to change this to the 64-bit dll libbsa64.dll, otherwise you will find the following error in your onbar log file:

(-43075) Unable to open file C:\ISM\2.20\bin\libbsa.dll.

2.3.10 Unsupported Onconfig Parameters

The following values are not required on Windows and can be discarded when converting an onconfig file from UNIX to Windows:

FASTPOLL – Fast Polling is a UNIX feature; the network implementation on Windows uses Completion Ports and does not require a FASTPOLL setting.

DUMPGCORE/DUMPCORE – Only the shared memory dump option is supported on Windows.

OPCACHEMAX – This optical cache tuning parameter is not supported on Windows.

DIRECT_IO – This bypasses operating system caching on UNIX and is not required on Windows.

NOAGE – Disabling priority aging is not supported on Windows. (Starting in IDS version 11, the NOAGE setting is included as an option in the VPCLASS parameter but it is ignored on Windows operating systems.)

2.4 Additional Programs Bundled with IDS for Windows

When IDS is installed on Windows operating systems, some platform-specific programs are included in the distribution. These programs are listed below with a brief explanation of their usage:

- **Copyboot.exe**
  Storage Manager recovery utility. Copies the emergency boot file from %INFORMIXDIR%\etc to a new name with "ixbar.servnum.date.time". This tool is used internally by scripts for building the sysadmin,

sysuser and sysutils databases.

- **Creates.exe**
  Creates a service. Useful if manually reconstructing an IDS instance. Input parameters are displayed if *creates* is called with no arguments. Note that the user name needs to be supplied with a computer or domain name (".\ " for a local machine) so *informix* would be written as .*informix*

- **Deletes.exe**
  Deletes a service. Takes the name of the service as an input parameter. Be careful with this program, it does not ask for confirmation.

- **Inssmnp.exe**
  Installs the Informix SNMP agent. Useful when you install the Windows SNMP monitoring service after you have installed the IDS instance because the Informix SNMP agent gets installed automatically only if the SNMP service is detected.

- **Instmgr.exe**
  The IDS Instance Manager, used to create, delete, modify, and rename IDS instances on Windows. It is discussed in more detail in the next section.

- **Ixpasswd.exe**
  Changes the informix password for all Windows services that log on as the informix user. See the Security Chapter 3.1 for an example.

- **Ixsu.exe**
  Launches a command shell running as another user (if no user name is provided the informix user is assumed). The functionality is similar to the Windows "runas" command, so ixsu is effectively obsolete in more recent versions of Windows.

- **Msgserv.exe**
  The IBM Informix Dynamic Server Message Service – runs as a Windows service and not a user-executable program.

- **Ntchname.exe**
  Changes machine name dependent entries in the IDS registry sub-keys. Use this program if you change the name of a Windows computer. This step is necessary for IDS to initialize successfully after a computer name change.

- **Ntmail.exe**
  A command-line utility to send email, comparable to a rudimentary version of the UNIX mailx command. Designed to be used by the IDS Alarm program (edit alarmprogram.bat to configure it) to send email to DBAs.

- **Onscpah.exe**
  The Online Service Control Program and Alarm Handler. This is the executable image for IDS Windows services. Not designed to be executed directly but launched when you start the IDS service.

- **Onsrvapd.exe**
  The server discovery process for SNMP.

- **Starts.exe**
  A utility to start an IDS instance from the command line. Typical usage: **starts %INFORMIXSERVER%**.

2.5. The Instance Manager

As IDS on Windows is implemented using a service, a program is bundled with the installation to ease the setup of new instances. This is the service instance manager.

The instance manager can be started either from its icon in the Informix program group or by typing **instmgr** in a command window where the informix path is defined.

After starting, the instance manager shows a list of the available instances on the computer. The instance manager can be used to create or delete an instance or to rename it.

When creating a new instance, the instance manager asks for the configuration parameters like service number, name and port and then creates the instance, defines the service, adds the registry keys, creates a command window with the right environment and asks whether the new instance should be initialized.

There is one instance manager per IDS version.

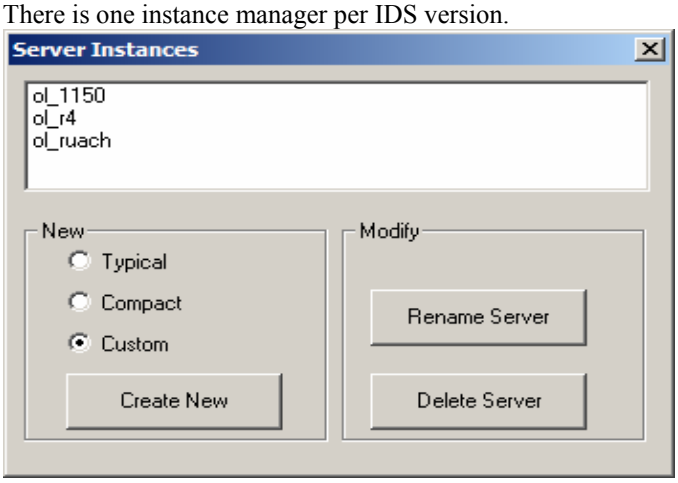

Figure 16 Server Instances window

If you start instmgr from the command line, you can add options to it. One option is *–system* to have the new instance run in the security context of the LocalSystem account.

3. Security

The IDS security subsystem has some platform-specific characteristics on Windows to match the fundamental differences in operating system security architecture between UNIX and Windows. This section looks at some frequently asked questions concerning IDS security on Windows and some implementation-specific details.

The most common areas of concern for IDS security on Windows tend to be the exploitation of known operating system vulnerabilities, and the risk of application-specific exploits such as buffer overflows. Both of these areas of concern can be offset by maintaining the most recent operating system and IDS interim versions, limiting remote access to the computer, and keeping user privileges and group membership to the minimum required levels.

Refer to the *IBM Informix Security Guide* for more information on administering Informix security features. Also, the IBM Redbook called *Security and Compliance Solutions for IBM Informix Dynamic Server* discusses security issues and covers topics such as encryption and role separation.

3.1 Issues Regarding the informix User on Windows

3.1.1 Changing the informix Password

As described above (see 1.1) the IDS instances and supporting services are implemented as services on Windows. If the informix user password is changed, the password for any Windows services that log on as the informix user must also be changed.

A utility called **ixpasswd.exe** in the %INFORMIXDIR%\bin directory can be used to change the password for every Windows service that logs on as the informix user. If executed with no arguments, the utility will prompt the user for the new informix password, and then scan the system for services that log on as informix.

Usage information is available with **ixpasswd.exe -?**. The tool can change passwords both for the local informix user and for a domain account.

Example:

```
C:\inf1000>ixpasswd

Changing password for services which log on as the local informix user

New password for Informix services:
Confirm new password:

Informix Service: marsh_1000
      Password changed for Service: marsh_1000

Informix Service: marsh_1150
      Password changed for Service: marsh_1150
```

3.1.2 Running Commands as the informix User

Some IDS commands require that the user either be informix or a member of the Informix-Admin user group.

The **ixsu.exe** utility is provided in *%INFORMIXDIR%\bin* to allow an administrator to launch a command session running as the informix (or any other) user by specifying the appropriate password. To use this utility, the administrator needs to have the following advanced user rights:

•Act as part of the operating system.
•Increase quotas.
•Replace a process level token

These rights are needed to be able to start a program as another user.

Alternatively, the Windows **runas** operating system utility can be used for the same purpose.

3.1.3 Running IDS as the Windows LocalSystem User

Starting with IDS 10.00.TC5, the IDS service can log on as the LocalSystem user. In fact it is no longer required that an informix user exist on the computer if the LocalSystem user is used. This allows system administrators to avoid password expiry issues with the informix user, and can be useful in circumstances where security policies do not allow a DBA to be a member of the Administrators group.

In IDS versions 10.0 and 11.10 the **setup.exe** and **instmgr.exe** programs have an undocumented command line argument *"-system"* that causes the informix user and group creation dialog boxes to be skipped, and the IDS service is installed to run in the context of LocalSystem.

In IDS version 11.50 LocalSystem user support is fully supported and documented. Running as a system user is an option during the Custom install process. A further option to bypass creation of the informix user is also offered.

The only caveat is that ER synchronization between UNIX and Windows is not supported if IDS is running as LocalSystem on Windows and no informix user exists on the computer.

If there is no informix user account on the system and you need to perform an administration task that can only be executed as the informix user, for example making a change to the sysadmin database, it is possible to signal the oninit process to execute the task in the context of the informix user. Oninit listens on a named pipe and will execute commands sent to that pipe by a member of the Administrators group. If you want to take advantage of this feature, you can get a utility called **onpipe.exe** from IBM technical support.

3.1.4 User Rights

Because the oninit process runs as a service on Windows and can perform privileged operations such as running processes as other users (in the specific case of the Stored Procedure SYSTEM command), it requires certain advanced user rights.

The local SYSTEM user has all required rights; but in the default case of the IDS service logging on as the informix user, informix requires the following advanced user rights in addition to the standard Administrator user rights:

- Act as part of the operating system
- Log on as a service
- Increase quotas
- Replace a process level token
- Debug programs
- Manage auditing and security Log

The IDS installation program sets these rights when the informix user is created, so it would only be necessary to manually assign these rights if the informix user were to be deleted and recreated or there was a pre-existing informix user present at install time who did not have these privileges.

It is possible to restrict specific user rights for users on member computers at an Active Directory Server level. If any of these rights are restricted for the informix user IDS will not function correctly.

3.1.5 Case Sensitivity

The user name “informix” is case sensitive. The IDS install utility creates the informix user with lower case letters. If you manually create an informix user make sure that the name is lower case.

SQL keywords are case insensitive. SQL data is case sensitive. A string-based query will need to either specify the same case as the data, use the upper() function, or define specific GLS files to implement case sensitivity.

File names are case insensitive. You do not need to be concerned about the case used when creating log files, dbspaces, and so on.

3.2 Windows Service Packs

Informix Dynamic Server is tested with the latest available Windows service pack level for each supported Windows platform before the product is released.

To minimize vulnerability to security exploits, always apply the latest Microsoft Service packs and hot fixes to all Windows computers in your network. Ensure that every service pack or hot fix is tested on a development computer before applying it to a production computer running IDS.

Always create a restore point so the fix can be rolled back and, if necessary, IDS can be restored.

In the unlikely event that a Windows service pack or hot fix causes an IDS problem, the problem is assigned a critical priority within IBM Support and Development to provide a fix in the shortest possible time.

3.3 Firewalls

By default IDS on Windows listens on a TCP port for incoming connections. The port number is set in the SQLHOSTS key of the Windows Registry:

HKEY_LOCAL_MACHINE/Software/Informix/SQLHOSTS/%INFORMIXSERVER%/->Service

The "service" can be a port number or a reference to a port number defined in the services file (%SystemRoot%/system32/drivers/etc/services).

If a firewall application is used on a computer that has IDS installed, the IDS listening port must be opened to allow incoming connections from any zone containing client computers.

If ISM is used, the ISM services require certain ports to be open for the *localhost* – typically 618 and 7938. These ports do not need to be open to external computers.

3.4 Virus Checking

On production data server computers, exclude chunk files from virus checking for the following reasons:

- When IDS is online, it requires exclusive access to chunk files. The use of a virus checking utility can interfere with this access.
- There is a small chance that binary data stored in a chunk could generate a false positive for a virus scanner that searches for known virus signatures. If this were to happen, the subsequent quarantine of the file could interfere with database availability.

- The CPU and disk resources used by a virus scanner to search a dbspace file are being wasted and could serve other purposes such as database server performance.

You can include other IDS files in a virus scan. If a virus scanner runs on your computer, include the IDS executable and DLL files in the scan.

If database performance is a priority, the CPU, memory, and disk resources used by a virus scanner should be considered. Some anti-virus products are more resource-intensive than others. Make sure database availability and performance is tested while a virus scanner is running before going into production.

To obtain maximum database performance consider implementing anti-virus security by other means that do not compromise performance, such as:

- Isolating the database server behind a firewall that limits connectivity to database access.
- Limiting use of the computer to applications directly connected with the database server.
- Limiting physical access to the computer.
- Restricting user with logon rights to a minimum.
- Shutting down any unnecessary Windows services.

3.5 User names

Currently the maximum Windows user name length supported by IDS is 20 characters – this is for backward compatibility with pre-Active Directory server domain configurations.

3.6 IDS Windows Security Implementation Differences

IDS security implementation on Windows operating systems has some advantages over those on UNIX. This section looks at some of these implementation differences.

3.6.1 File Permissions

Access to files and other operating system objects such as registry keys is controlled through Discretionary Access Control Lists (DACLs) on Windows. At install time, IDS sets DACLs on its directories to prevent users who are not members of the Informix-Admin user group from writing to them. Similarly, IDS registry keys are protected with DACLs to prevent accidental or malicious corruption.

Though different in implementation and usage, from an IDS perspective, DACLs provide similar capabilities to UNIX user/group/other file permissions.

As it is important to ensure the security of both the installation files and the data, System Administrators should regularly check the access permissions to these files. There are utilities available with the operating system, like **attrib**, **cacls** and **icalcs**, which let you display and modify file attributes. Other utilities, such as **xcalcs** and **perms**, are part of the resource kit. There are also third party products available, which can be used to grant rights.

3.6.2 Symbolic Links

On UNIX an area of security concern is the misuse of symbolic links by malicious users linking to known file names in unprotected directories. The Windows equivalent of symbolic links – Hard Links and *Junction* or *Directory Reparse* points - tend to be less well known but are also potential candidates for exploitation. For this reason, it is important to verify that access control to Informix directories is limited to trusted users and to make sure that log files and other user-configurable paths go to protected directories.

4. Performance

Informix Dynamic Server performance tuning applies to all platforms. A well-tuned application on UNIX will also perform well on Windows. The References section of this document has a link to an IBM White Paper on generic IDS performance tuning tips.

This section will look at some of the platform-specific performance considerations for the Windows platform.

4.1 Baseline Information

One core consideration for performance tuning is to get baseline information, to know what the system is doing when not having performance problems. This information will give you a reference to compare against.

Basically, such a baseline should consist of performance information both from the operating system and from the database instance. The information should cover processes and CPU usage, memory usage, disk I/O, and network performance.
To get the operating system information, you can use the Performance Monitor included with Windows. It can be used to do on the fly performance monitoring, but can also be configured to collect information in specific intervals and write those to a log file.
It is available from **Administrative Tools**->**Performance**. A sample log file can be found in the Counter Logs tab.

If you want to access the performance counters from a command-line script, the counters are exported using the Windows Management Instrumentation (WMI) infrastructure, so that any programming or scripting language with an interface to WMI can use them.

On IDS you can get information about the same components by using **onstat** commands. Basically, you would look for **onstat –g glo**, **onstat –g sch**, **onstat –g rea** (just count the number of threads in the ready queue for CPU usage), **onstat –g mem** for the memory, **onstat –g ioa** for I/O and **onstat –g nta** for the network information. How often you use the **onstats** utility depends on the way your system is used.

You can also use the OpenAdmin Tool (OAT) to get some of the baseline information. OAT is also useful for identifying what kind of queries are running slow on your system. You might want to save query plans for standard queries. These plans are useful if you find that the behavior of a query has changed (for example, after an update).

4.2. Specific Performance Considerations for Components

4.2.1 Process and CPU Usage Concerns

- Look at all user and system processes running by using the Windows Task Manager or Sysinternals Process Explorer and understand the purpose of each process. Disable any processes that are non-essential for operating system and data server operation.

- Measure the performance attenuation caused by virus scanners and other security processes on computers running IDS and favor solutions such as firewalls to minimize access. See also section 3.4.

- Windows "client" operating systems such as Windows XP and Vista are configured by default to optimize performance for user processes running in the foreground rather than background server processes. See the references section for resources to assist in maximizing performance of server programs.

4.2.2 Memory Usage

- Use the /3GB boot.ini switch for 32-bit Windows platforms with available physical memory.

- For applications with large memory requirements, determine how to maximize available shared memory, particularly for IDS on 32-bit Windows computers. See section 1.4 and 5.3 for more details.

- Computers with sufficient memory (~2 GB or more) can take advantage of the improved caching algorithms and TCP stack performance improvements in Windows Vista and Windows Server 2008.

4.2.3 File System and I/O Subsystem Concerns

- Tune IDS caching and memory ONCONFIG parameters for 4 KB page size (or use configurable page size and tune accordingly). Examples include BUFFERPOOL, RA_PAGES / RA_THRESHOLD, LOCKS. See section 2.3.

- Monitor disk fragmentation and de-fragment if necessary. Excessive file system fragmentation has a negative effect on general Windows performance. Any fragmentation on dbspace cooked files will affect IDS performance. Once allocated, a chunk should not become more fragmented.

  For Windows Vista and Windows Server 2008, consider scheduling defragmentation tasks to coincide with times when data server demand is at a minimum because I/O performance will be reduced when any defragmentation is taking place. Do not defragment IDS chunk files when the instance is online.

- Consider the performance implications of temporary dbspaces and setting DBSPACETEMP. Sorts can run slower in temporary dbspaces – see section 1.3.1.

4.2.4 Networking and Security

- For IDS domain installations, consider the network overhead of authenticating connections with Windows Domain Controllers. Avoid network topologies that require authentication over a WAN.

- User authentication can take longer for remote users in a large domain forest. Consider creating local users for database access or implementing connection pools to reduce the number of connects/disconnects.

- In IDS version 10.00TC5 or later, if a user connects without specifying a domain, only the local machine and primary domain are checked. If you want to check for the user name in all trusted domains, you can add CHECKALLDOMAINSFORUSER 1 to your onconfig file.

- For connection pools and other applications where a large number of users log on at the same time, consider increasing the number of MSC VPs to service the operating system authentication calls. This overhead can become more pronounced if PAM authentication is used.

- Consider DNS lookup overhead, check the binding order of the Windows network adapters and favor TCP/IP binding. Remove any unneeded network protocols.

- Use the latest versions of Informix drivers such as ODBC, JDBC, ESQL/C, and check both the Client SDK and IDS release notes. Recent ODBC driver fixes, for example, resolved performance problems in buffer size and sending VARCHAR data types over the network.

5. Usability

This section describes Windows specific settings relevant in deploying IDS.

5.1 System Command Usage

5.1.1 System Command in Stored Procedures

When calling a system command within a stored procedure, a process is created for whatever argument is passed to the system command.

The process that is created runs with the user and access tokens of the database client user who is calling the stored procedure. The **oninit** process needs to log on as that user, and therefore needs access to the user's password. The **oninit** process will have access to this password only if the user connected to the database with a user name and password. Local database users are not forced to connect with a password and if they do not, a stored procedure SYSTEM command will fail with a 668 error.

```
Cannot execute stored procedure start_onpload SQL ERROR -668
```

To prevent this error, you can store the user password with **setnet32** or specify the user password at connection time.

5.1.2 Controlling SYSTEM Command Desktop Interaction

When using the system command in a stored procedure, the new process is created with the option to be in the foreground and to interact with the desktop.

If system commands are being executed frequently they can cause problems for users logged on to the desktop because command windows will pop-up each time a system command is run.

You can prevent interaction with the Windows desktop when a Stored Procedure Language (SPL) routine executes a SYSTEM command by setting the environment variable **INTERACTIVE_DESKTOP_OFF** to 1.

If INTERACTIVE_DESKTOP_OFF is set to 1 and an SPL routine attempts to interact with the desktop, for example if **notepad.exe** was executed, the routine will fail unless the user is a member of the Administrators group.

The valid settings (1 or 0) have the following effects:

1 - Prevents the database server from acquiring desktop resources for the user executing the stored procedure.

0 - SYSTEM commands in a stored procedure can interact with the desktop. This is the default value.

Setting the INTERACTIVE_DESKTOP_OFF environment variable to 1 also allows an SPL routine that does not interact with the desktop to execute more quickly because internal function calls to acquire desktop handles can be bypassed. In addition, a greater number of SYSTEM commands can be called at the same time because the command no longer depends on limited operating-system resources (Desktop and WindowStation handles).

The INTERACTIVE_DESKTOP_OFF environment variable must be set in the environment of the IDS instance that you are using. Define it either as a Windows system environment variable or add it to the Environment key of an instance: HKEY_LOCAL_MACHINE\SOFTWARE\Informix\OnLine\<instance name>\Environment

5.1.3 Where to Set Environment Variables

You can set environment variables in several places on Windows, depending on which IBM Informix application you use.

If you need to set the environment for an IDS Instance, you have the choice of setting variables with **setnet32**, adding System or User environment variables in the Control Panel, or setting variables at the command prompt (this would only take effect if the **oninit** process was launched in the foreground and not started as a service).

When the IDS service is started, any settings defined in the
HKEY_LOCAL_MACHINE\SOFTWARE\Informix\OnLine\<instance name>\Environment
will be read and will override other values.

5.2 Running HPL on Windows

On Windows there is no graphical interface for running the High Performance Loader (HPL), but the **onpladm** utility can be used to manage HPL.

The **onpladm** utility uses a stored procedure in the sysmaster database to start onpload jobs with SYSTEM commands.

For example, to create a project to unload the stores_demo database you could use:

onpladm create project p_stores –d d:\data –D stores_demo

Then run the project:

onpladm run project p_stores –fu

Set the INTERACTIVE_DESKTOP_OFF environment variable to avoid seeing command windows pop up for every started job. The **onpladm** utility requires you to be connected to the database with a user name and password. See 5.1.1 for details.

5.3 How to Set Up RSS on Windows

This section describes the steps needed to set up an RSS system.

Ensure that the connectivity is working between both computers and that you can restore the backup from the primary computer on the secondary computer. This might involve renaming the backup if you are doing ontape into directories. The naming convention used for backups is <*machine name*>-<*servernum*>-<*level of backup*>.

To check the connectivity between both computers, bring the RSS computer online and check that everything is working smoothly before applying the restore of the primary computer to it.
For this reason we discuss setting up the connectivity first, before discussing the steps to actually get the RSS working.

To ensure that connectivity is working between both computers, add the entries of both computers to both sqlhosts (that is, registries).

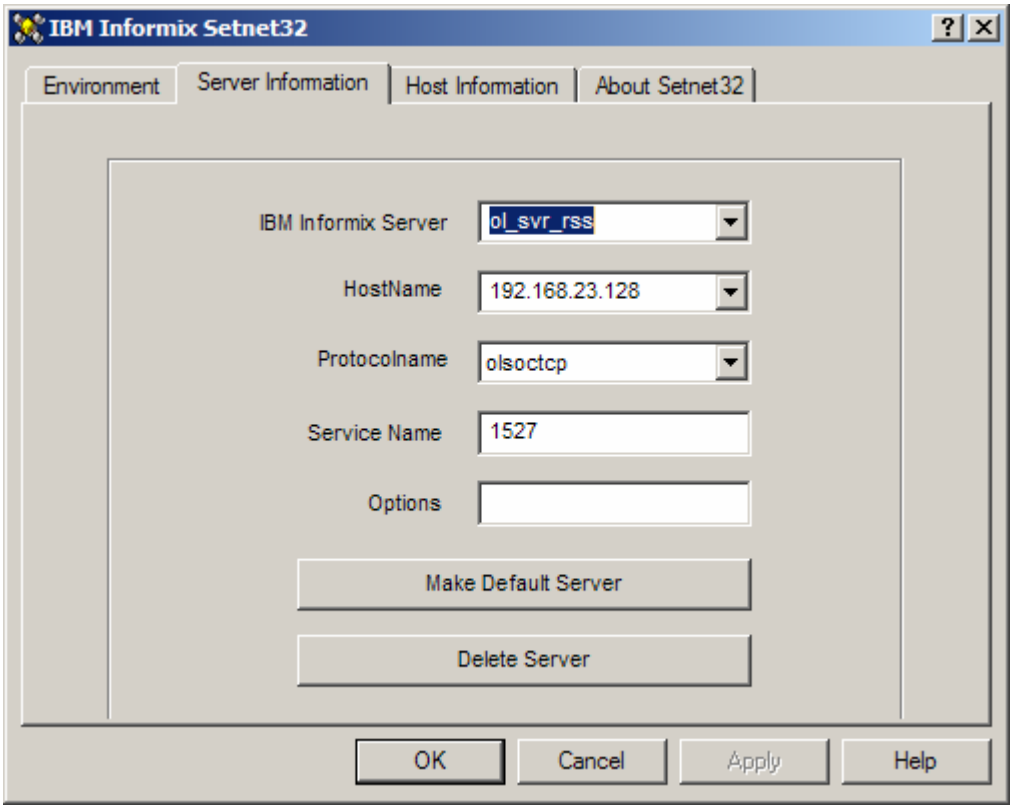

Figure 17 Server Information in Setnet32

Then check whether you can access the other computer with dbaccess. Verify that the Host Information is set on both computers using the PASSWORD settings for the user informix. You can check whether access is working without supplying passwords either by using ilogin demo or by running dbaccess as the user informix and trying to connect to the other computer without supplying credentials.

If connectivity is working fine, getting RSS setup is easy. We will assume here that the primary computer is called ol_svr_prim and the RSS computer is called ol_svr_rss.

First, you need to create the primary instance. Remember to set **LOG_INDEX_BUILDS** 1 in the onconfig file . Also make sure that you backup the logical logs. Then do a level 0 backup of your primary. After you have done this, you can start RSS on the primary with the **onmode –d add RSS ol_svr_rss** command

The online.log shows the following information:
11:46:44 Building 'sysha' database ...
11:46:45 Unloading Module <SPLNULL>
11:46:45 'sysha' database built successfully.
11:46:45 DR: Reservation of the last logical log for log backup turned on
11:46:45 RSS ol_svr_rss added

In the next step, apply the restore on the RSS computer. Remember that you have to do a physical restore only, so you have to start the restore with either the **ontape –p** command or the **onbar –r –p** *command.*
As mentioned above, if using ontape into directories you will have to rename the backup file. The ontape utility will print out the name of the file it wants to restore from. Take into account that on Windows the chunks are usually created in <drive>\IFMXDATA\<instance-name>. If ol_svr_prim has followed this convention, you need to make sure that the onconfig for ol_svr_rss points to the right rootdbs.

Once the restore is done, you can start RSS on the secondary as well, using the **onmode -d RSS ol_svr_prim** command.

The online.log should than show similar information to this output:

```
12:36:00  DR: new type = RSS
12:36:00  DR: Owner of the disk is set as ol_svr_rss.
12:36:00  RSS Server ol_svr_prim - state is now connected
12:36:01  Logical Recovery Started.
12:36:01  10 recovery worker threads will be started.
12:36:01  Dynamically allocated new virtual shared memory segment (size 8192 KB)
12:36:01  Memory sizes:resident:45504 KB, virtual:40896 KB, no SHMTOTAL limit
12:36:01  Start Logical Recovery - Start Log 2, End Log ?
12:36:01  Starting Log Position - 2 0x9c2018
12:36:01  Clearing the physical and logical logs has started
```

RSS is working between both computers.

5.4 How to Install Connection Manager on Windows

The Connection Manager dynamically routes client application connection requests to the most appropriate server in a high-availability cluster. To do so, the Connection Manager needs to maintain connections to each server in the cluster.

The Connection Manager is implemented as a service on Windows, to use the advantages of the service architecture as described in Chapter 1.1.

To set up the Connection Manager on Windows it needs to be configured with the connection information about these servers. This means adding the information into sqlhosts (that is, the registry). Additionally the Connection Manager needs a configuration file cmsm.cfg , which should reside in %INFORMIXDIR%\etc. The Connection Manager is part of the Client SDK, so you will find an example for the configuration file in the %INFORMIXDIR%\etc path used by Client SDK, the default path is C:\Program Files\IBM\Informix\Client-SDK\etc.

The cmsm.cfg file defines the name of the Connection Manager and the Service Level Agreements. A simple cmsm.cfg file for the primary and rss configuration from Chapter 5.3 could look like this.

```
NAME wincm
SLA oltp=ol_svr_prim
SLA report= (ol_svr_prim + ol_svr_rss)
```

Although the LOGFILE setting for Windows suggests using c:\informix\tmp for the log file, use the %INFORMIXDIR%\tmp settings for either Client SDK (you would need to create a temp directory there) or the Server.

Now you need to add the name used in the SLA to the sqlhosts file.

The following figure shows the setnet information for the report group, a similar entry must be added for the oltp group.

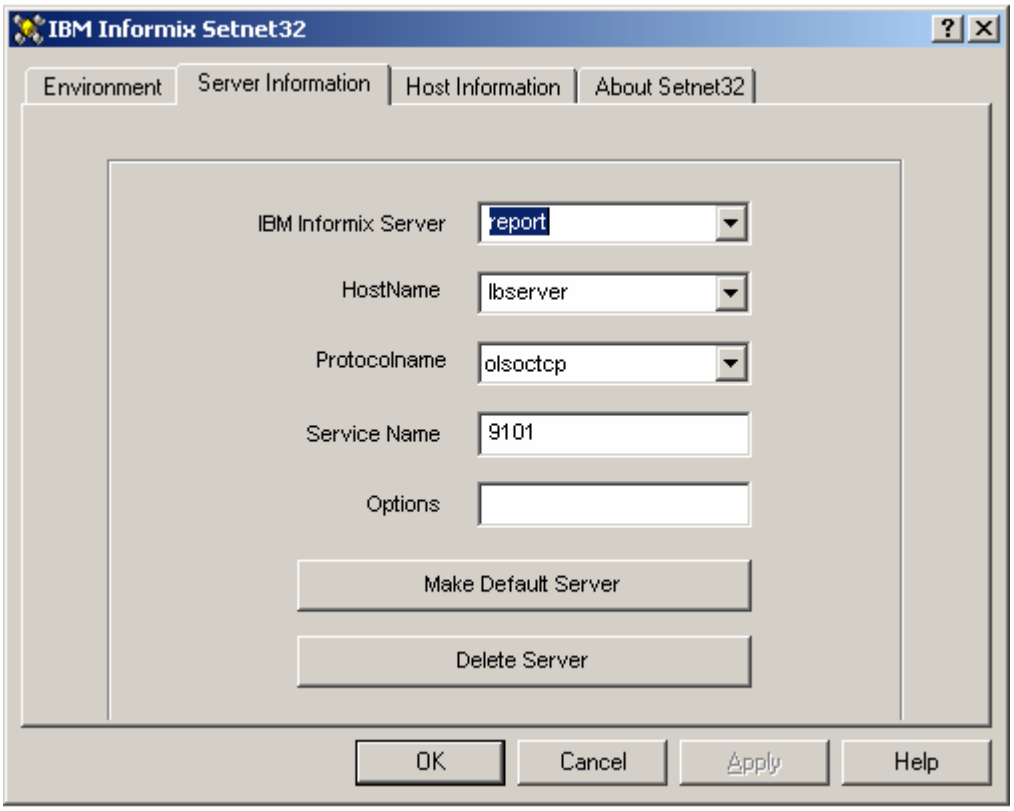

Figure 18 Setnet information for a Connection Manager entry

Next, you can install the service with **oncmsm –i.**. The %INFORMIXDIR% must point to your Client SDK installation, and you must have %INFORMIXSERVER% defined.

In IDS 11.50.xC2 and later you have the option to install Connection Manager either by using LocalSystem or by entering a user name and password. In the following example, we use user informix, because this user already has the appropriate rights and can connect to all the servers needed, as defined in 5.3.

```
C:\Program Files\IBM\Informix\Client-SDK\bin>oncmsm -i
17:21:04 IBM Informix Connection Manager log file: C:\tmp\cm.log

Please specify the user and password to run this service.
Press <ENTER> to run Connection Manager as 'LocalSystem'
User name: informix
Password:
Confirm password:
Connection Manager Service "wincm" installed successfully
```

Next start the service with **oncmsm**. The cm.log should show something similar to the following excerpt:

```
17:22:09 IBM Informix Connection Manager
17:22:09 Connection Manager name is wincm
17:22:09 Starting Connection Manager...
17:22:09 Warning: Password Manager failed; working in trusted node mode
17:22:09 dbservername = ol_svr_prim
17:22:09 nettype = olsoctcp
17:22:09 hostname = RUACH
```

17:22:09 servicename = turbo5
17:22:09 options =
17:22:09 listener oltp initializing
17:22:09 listener report initializing
17:22:09 Listener oltp=ol_svr_prim is active with 16 worker threads
17:22:09 Listener report=(ol_svr_prim+ol_svr_rss) is active with 16 worker threads
17:22:10 Connection Manager started successfully

You can test the Connection Manager with dbaccess. Go to the connection menu in dbaccess to list the new servers oltp and report. Selecting a connection to oltp results in a connection to ol_svr_prim. Selecting a connection to report results in the client connecting either to ol_svr_prim or to ol_svr_rss.

The status of the Connection Manager can be monitored with **onstat –g cmsm**.

IBM Informix Dynamic Server Version 11.50.TC3 -- On-Line -- Up 00:04:24 – 78208 Kbytes

CM name	host	sla	define	foc	flag	connections
wincm	RAUCH	oltp	ol_svr_prim	SDS+HDR+RSS,0	3	1
wincm	RAUCH	report	(ol_svr_prim+ol	SDS+HDR+SSS,0	3	1

5.5 Windows Vista and Server 2008 Compatibility Issues

5.5.1 IDS Support Status

The following IBM Informix products are supported on Windows Vista:

- Informix Dynamic Server 11.10 and later
- Client SDK & I-Connect 3.00 and later

Note that Windows Vista Home Edition is not supported.

The following product versions are supported on Windows Server 2008:

- Informix Dynamic Server 11.10.xC3 and later
- Informix Dynamic Server 11.50.xC3 and later
- Client SDK & I-Connect 3.50.xC3 and later

Other versions of IDS, such as version 10.0, will run on Vista and Server 2008 without official support. Some minor compatibility problems have been observed (see section 5.6.3).

5.5.2 Reasons to Run IDS on Windows Vista and Server 2008

Windows Vista is intended to be a “client” operating system - an upgrade for Windows XP - rather than a “server” operating system such as Windows Server 2003. For server machines the logical upgrade path is Windows Server 2008.

IDS is expected to perform better on “server” operating systems out of the box due to networking and scheduling configuration being tuned for server processes.

Reasons to use the more recent Windows operating systems for IDS are related to performance and security.

Performance

If a computer has sufficient memory it can make use of improved caching algorithms in Vista such as SuperFetch. The TCP/IP stack also has improved network performance, particularly for computers with multiple CPUs. Automated maintenance tasks such as defragmentation can help maintain system performance over time.

Tests of a simple SQL fetch on a dual-core computer with 2 GB RAM showed Vista to be 18% faster with 500 users and 30% faster with 1000 users compared to identical hardware running XP. The same improvement was also measured comparing IDS running on Windows Server 2008 to Windows Server 2003.

These tests did not simulate real applications but do imply a potential benefit on computers with sufficient memory.

Security

The many security enhancements in Vista, such as improved out of the box security, encrypted file system support and UAC, are potential benefits to moving to this platform.

5.5.3 Vista and Server 2008 Incompatibilities Related to the User Account Control Feature

The User Account Control (UAC) feature of Windows is designed to minimize the use of the Administrator privilege by requiring all users to run without Administrator privileges unless a program requests elevated privileges.

The UAC feature does not apply to Windows services so the IDS service is unaffected by this feature. However, user programs that start a service or write to the HKEY_LOCAL_MACHINE registry hive must run as administrator.

In IDS 11, Administrator users will see the following prompt if they execute a program such as **setnet32.exe** or **instmgr.exe**, which require elevated privileges:

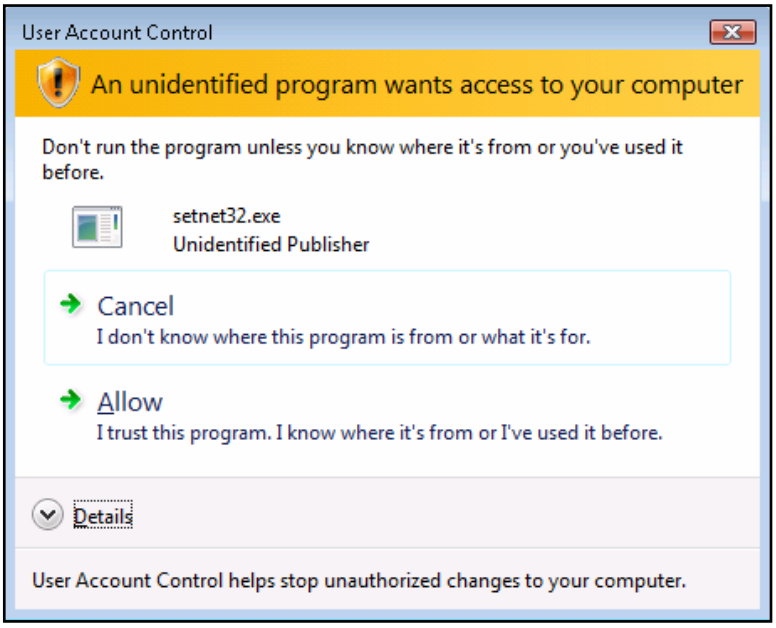

Figure 19 Windows prompt for a process set to run as Administrator

Other programs that are registered with Microsoft will show this prompt:

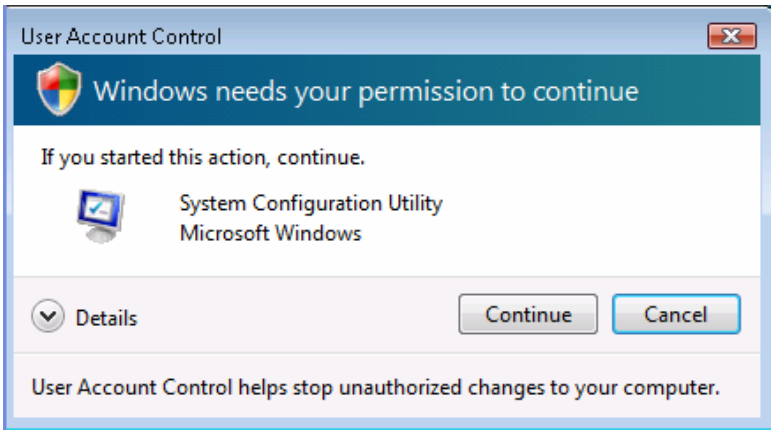

Figure 20 Windows prompts for a trusted program asking to run with elevated privileges

In earlier versions of IDS, the user has to manually run setnet32 and instmgr as administrator by right-clicking on the program icon and selecting "Run as administrator".

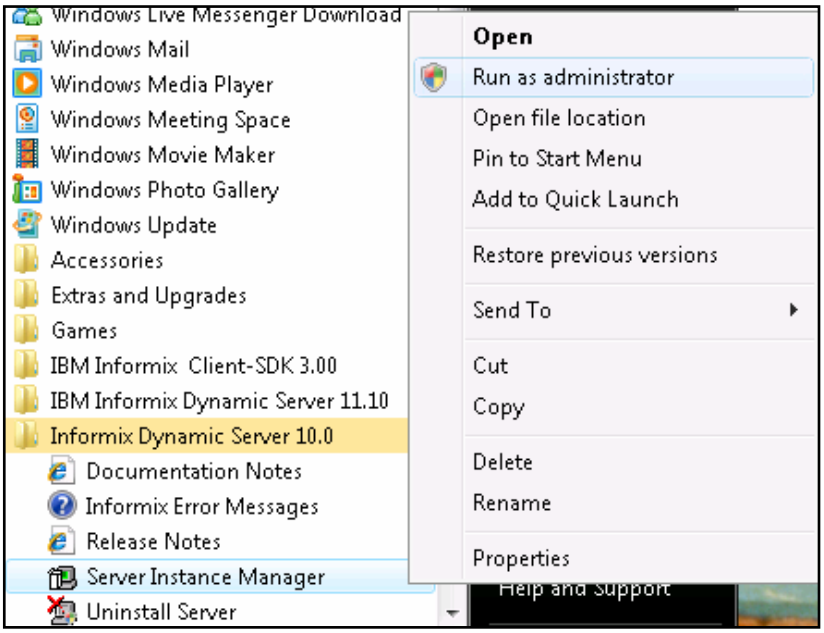

Figure 21 Running a program as administrator

Alternatively the Properties option can be selected, allowing the program to be configured to always generate the "Run as administrator" prompt.

On systems that have UAC enabled, run command-line programs that need UAC rights from an administrative command line. Start the instance command-line window with "Run as administrator".

If the setnet32 program is run in unprivileged mode it will work correctly for most operations, such as setting client environment variables, but will fail if an action causes it to attempt to write to the HKEY_LOCAL_MACHINE section of the registry such as defining a new server. The resulting setnet32 error is "You cannot modify Environment settings for Local Machine. Want to reload environment from Local Machine?"

5.5.4 Disabling the User Account Control Feature

One way to avoid application compatibility problems associated with the Windows UAC feature is to turn off the feature by running the **msconfig.exe** Windows utility and selecting the "Disable UAC" tool. This will cause the User Account Control feature to be switched off when the system is restarted.

UAC is an important security component but administrators might choose to disable it on systems that are sufficiently isolated (such as test systems) or that are made secure by other means.

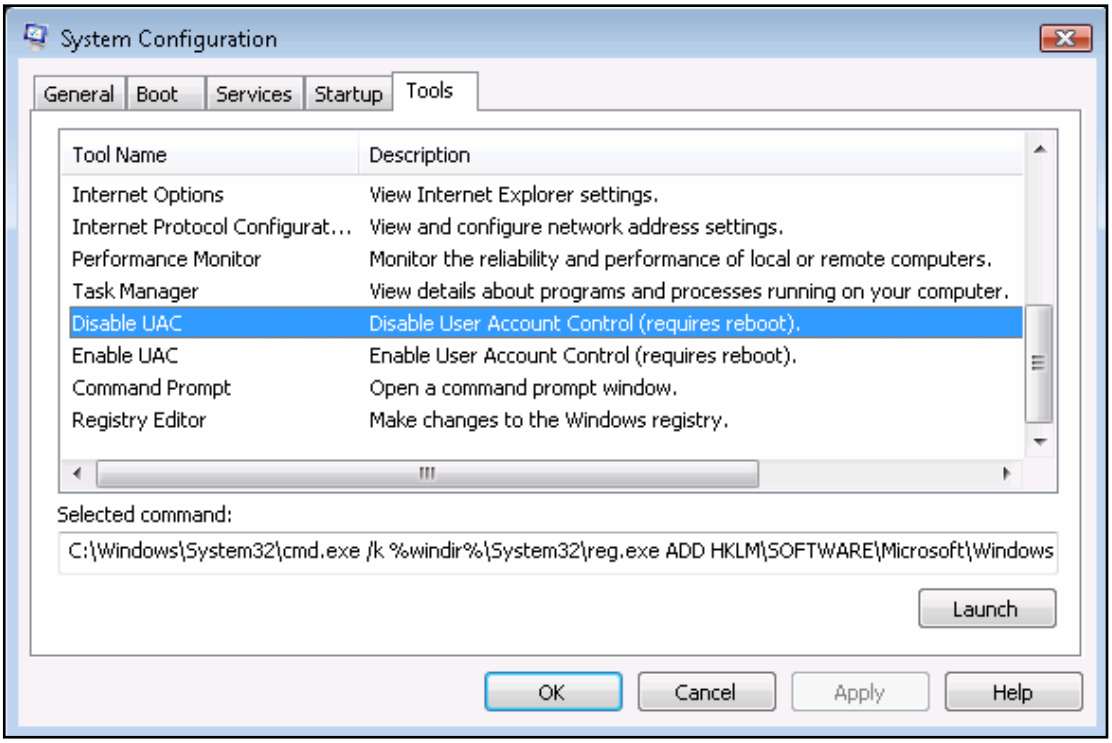

Figure 22 Running the Msconfig utility to disable UAC

On Windows Server 2008, the UAC settings can be found in the Security Options in the Local Security Policy applet. There are multiple settings, which allow you to granularly define the level of security that you want.

If you want to eliminate the prompt when using administrative rights you can select "elevate without prompting" for the administrators. However, command-line programs will still need to be started from a console that was started as administrator. For more details, refer to the Microsoft documentation.

6. Troubleshooting

For the majority of IDS problems, the same troubleshooting techniques can be followed for all platforms. This section discusses the major areas where troubleshooting differs on Windows operating systems.

6.1 IDS Initialization Problems

6.1.1 Run oninit.exe in the Foreground

On UNIX operating systems, one way to troubleshoot oninit initialization problems is to start IDS in verbose mode using the *oninit –v* command. Progress messages are then printed to the console during initialization.

On Windows operating systems, because the oninit process runs as a service and does not have a console by default, if the verbose parameter is used the output is lost. One way to view the verbose output is to start the oninit process in a command window, bypassing the service environment. To do this start the IDS command window as the informix user, and run **oninit.exe –v**.

The oninit process will print its verbose output to the command-line window. Because the oninit process does not fork and exit as it does on UNIX, if initialization does not fail, oninit will run until the *onmode –k* command is executed from another window. Closing that command window will also kill the instance.

6.1.2 Run oninit as a Service in Verbose Mode

If the **oninit.exe** process initializes successfully in the foreground but fails if IDS is started as a Windows service, it can be useful to start oninit in verbose mode as a service. Setting the operating system environment variable ONINIT_STDOUT to the path of a file with the System Applet before starting the IDS service will cause any messages to **stdout** or **stderr** to be printed to this file once IDS is re-started.

For example, suppose the ONCONFIG value in the IDS instance registry Environment key is set incorrectly. The service will fail to start with no messages to the online log, and if started with the Services Applet a pop-up message will appear: “The Informix IDS – ol_1150tc3 service on Local Computer started and then stopped. Some services stop automatically if they have no work to do, for example, the Performance Logs and Alerts service.”

If **oninit.exe** is started in the foreground it will start correctly as long as %ONCONFIG% is set correctly in the command-line environment.

If the ONINIT_STDOUT environment variable is set to *c:\temp\oninit.txt* and the IDS service is started with the *-v* argument, for example by running “starts %INFORMIXSERVER% –v”, the oninit.txt file will contain the following information:

```
Checking group membership to determine server run mode...succeeded
Reading configuration file 'c:\inf1150/etc/xxxxx'...FAILED
```

This message provides sufficient information for the cause of problem to be identified.

6.1.3 Monitor oninit Initialization Sequence Using Process Monitor

If you suspect that IDS initialization problems are related to accessing information in the registry or in a file, use the Windows Sysinternals utility Process Monitor to monitor which registry keys and values, or which files, the **oninit.exe** process is reading**.**

Process Monitor monitors specific key events: accessing the registry, accessing the file system and starting and stopping processes. It is a superset of the Windows Sysinternals Regmon and Filemon tools.

You can set filters to monitor specific processes and define what you want to see. A key feature of Process Monitor is that the filtering is not destructive; the underlying data will be preserved, so that if you have filtered away too much, you can try again.

To monitor IDS initialization the following "Process Name begins with" filter will suffice:

oninit;onscpah

This includes the Online Service Control Program and Alarm Handler that interfaces with the Windows Service Control Manager as well as oninit.

Process Monitor also supports other options that allow filtering with conditions such as path names, operations, and parent process.

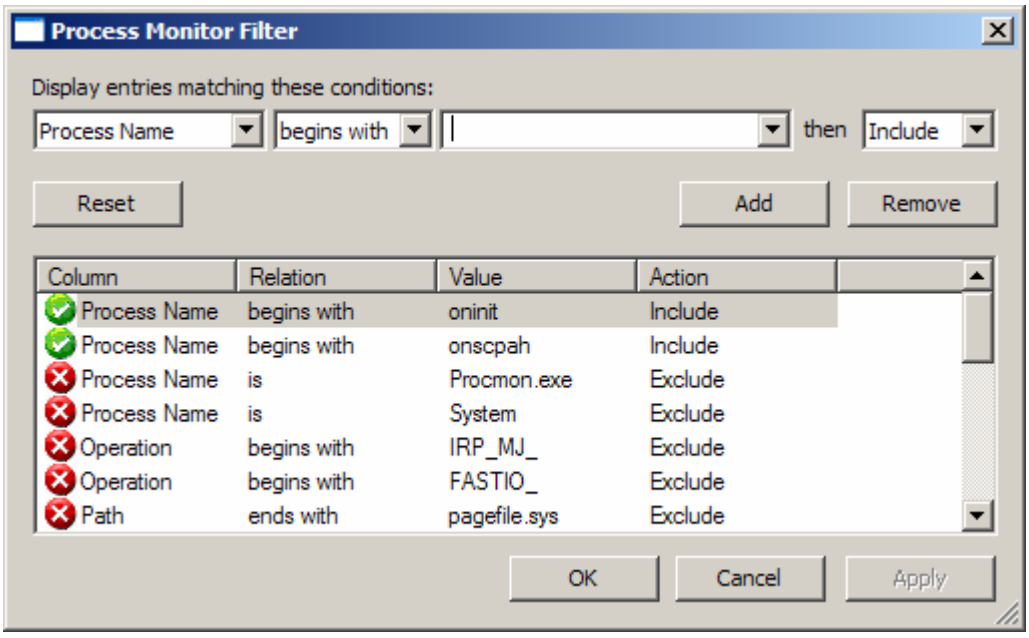

Figure 23 Setting up a Process Monitor filter for IDS

Once IDS is started, file and registry access attempts and access results are listed in the Process Monitor GUI, as are process-creating activities.

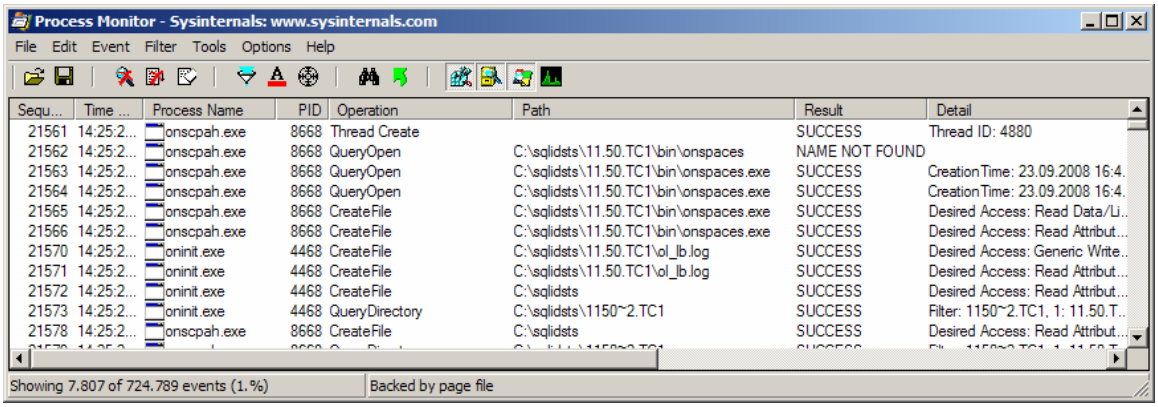

Figure 24 Part of Process Monitor output for IDS startup

Use the Process Monitor output to examine, registry key read failures or other problems and check them for possible registry corruption.

Process Monitor can also be used to analyze file access problems, or problems during process creation.

The Registry Summary in the Tools menu shows which Registry keys were accessed from the entries selected by the current filter. A similar view exists for files in the file system. The summary provides a helpful overview of which keys or files are accessed.

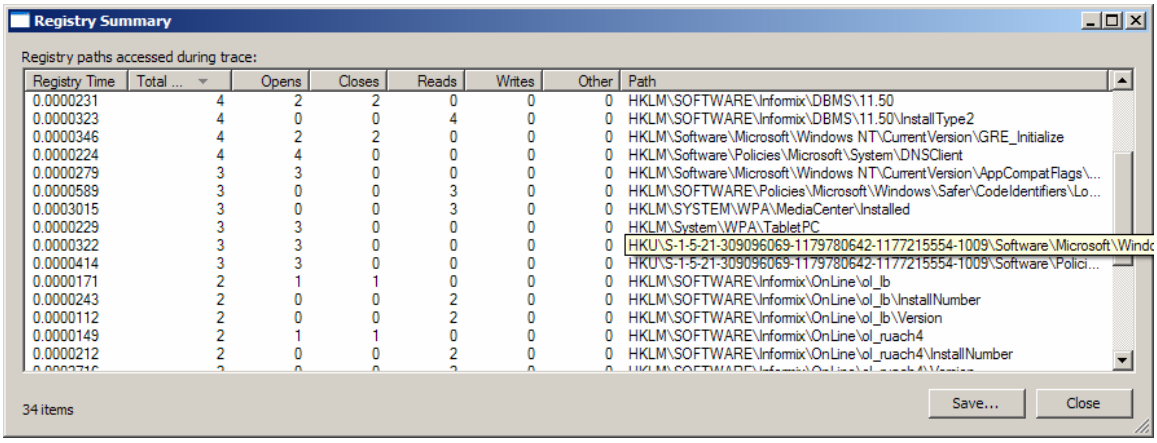

Figure 25 Registry Summary Window

6.1.4 Using the Log File for the IDS Services

The IDS instance services write to a log file in %Systemroot%\system32 called **olsrvice.log**.
During normal operation the only events recorded there will be events such as "*Service was shut-down externally"*.

Any Windows errors during the startup of the server are also described in that log. For example:
ol_1110: Failure in SuCtrlHandler: CreateThread(): 1455: The paging file is too small for this operation to complete.

In cases of startup failures the name of the service will be added to the olsrvce.log information.

6.2 Resource Leaks

6.2.1 Monitor Oninit Process Handle Count

A common cause of resource leaks in Windows programs is process handle counts that keep growing. A handle is a unique descriptor that represents operating system objects. Open files have handles (analogous to UNIX file descriptors) as well as processes, threads, and other resources.

If a handle is created by a process and not closed, the process handle count increases. If the handle count continues to grow, the process can exhaust its available working memory (Working Set) and fail. Symptoms of this failure might include errors that occur every few days and disappear after the process is re-started.

The current version of IDS has no known resource leaks at the time of writing, but it is useful to know how to diagnose this problem, in oninit and any application processes that communicate with the data server.

The simplest way to monitor the handle count on Windows is to launch the Windows Task Manager, select the **View->Select Column**s menu option, and select **Handle Count** from the *Select Columns dialog box*.

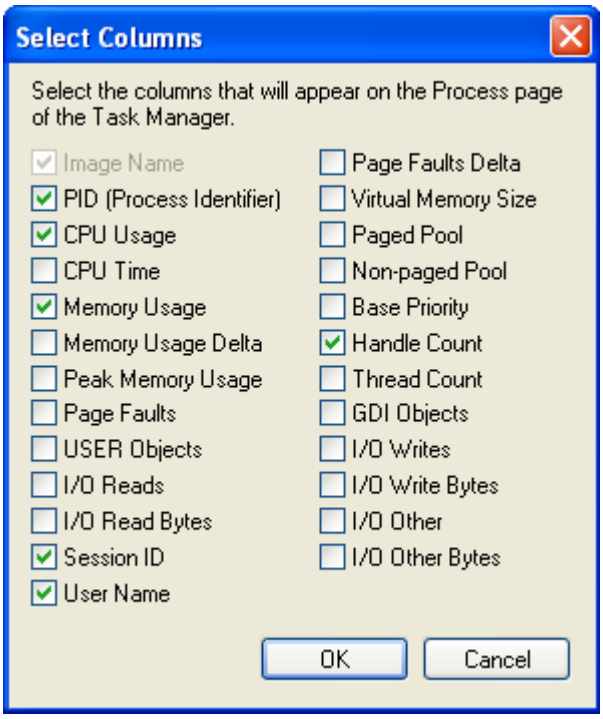

Figure 26 Windows Task Manager Select Columns dialog box

The handle count for a process can then be monitored from the Task Manager. An increase in the handle count indicates a potential resource leak that should be reported to IBM Technical Support.

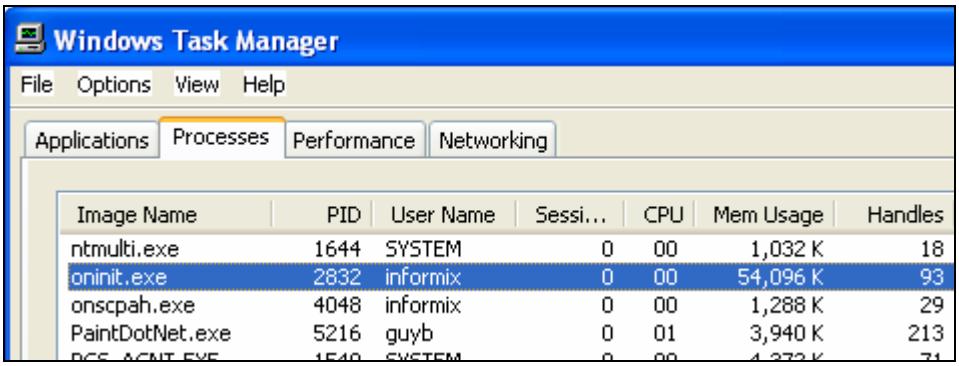

Figure 27 Windows Task Manager window showing handle count

Handle counts can also be monitored with the Process Explorer.

6.3 Shared Memory Initialization Problems

6.3.1 Addressing Memory Allocation on Windows Server 2003, SP 2

Anyone who likes to use large shared-memory configurations on 32-bit Windows has to think about where DLLs are loaded in memory by processes such as **oninit.exe**, **onmode.exe**, **onstat.exe** and **onbar.exe**. Any process that attaches to a shared-memory segment needs a matching contiguous range of address space, and any DLL loaded by that process potentially fragments the address space.

As discussed in Section 1.4.2, there are conventions for where DLLs should be located in the address space. However, there is a recurring issue with operating system DLLs not following that convention.

For Windows Server 2003 SP2 one known problem is that the home networking configuration manager DLL **hnetcfg.dll** is loaded at the following address 0x5F270000, which will fragment the available memory. This can lead to oncheck or onstat error messages such as the following one:

```
shmat: [22]: operating system error
MapViewOfFileEx: w32ec=487 at nt_shm.c:663
Unable to attach to shared memory.
Invalid argument
```

To resolve this problem, ask Microsoft for a fix for the following defect:
948656 A program that allocates a large block of contiguous memory may not start or may intermittently fail in Windows Server 2003

6.3.2 Measuring IDS Memory Requirements

With the following IDS 11.5 ONCONFIG parameters:

```
SHMBASE       0xC000000L
SHMVIRTSIZE  204800
LOCKS          800000
BUFFERPOOL size=4K,buffers=200000,lrus=16,lru_min_dirty=3.000000,lru_max_dirty=4.000000
```

The *onstat -g seg* output at startup on looks like this:

```
Segment Summary:

id          key         addr       size        ovhd      class     blkused    blkfree
1381386241  1381386241  c000000    922877952   245568    R*        225309     3
1381386242  1381386242  43020000   209715200   7048      V         3120       48080

Total:      -           -          1132593152            -         228429     48083

   (* segment locked in memory)
```

The relevant parts of this (converted to hex) are:

Address	Size	Class
0xC000000	0x37020000	R
0x43020000	0xC800000	V

The allocated space goes from 0xC000000 to 0x4F820000, and the two segments together have a total size of 0x43820000. This is small enough for a process to load both segments on an un-patched computer with some room left over, even if it were to also load xpsp2res.dll at 0x10000000.

6.3.3 Using Onbar with a Large Shared Memory Configuration

In versions before 10.00.xC7 or 11.10.xC2, you could run into a defect where onbar could not work with instances with large shared-memory configurations. Update to later version of 10.00 or 11.10.

6.3.4 Check Segment Addresses Against DLL Load Addresses

Sometimes a utility such as **onbar** or **onstat** will fail to connect to IDS shared memory with an error like this one:

```
MapViewOfFileEx: w32ec=487 at nt_shm.c:663
17:41:21 shmat: [22]: operating system error
onstat: Cannot attach to shared memory. errno = 22
```

A typical cause of this error is that the utility connecting to IDS shared memory is loading a DLL at an address that fragments its available contiguous addressing space leaving no room to map the IDS shared memory segment. See section 1.3 for a detailed discussion of DLL address space problems.

The standard troubleshooting approach adopted by Informix technical support is:

1. Record the addresses of the IDS shared memory segments by running *onstat –g seg*.

2. Reduce the size of IDS shared memory by lowering the BUFFERPOOL and LOCKS ONCONFIG parameters in a test instance until the problem goes away.

3. Run the utility that had the problem and map its DLL load addresses using Process Explorer (see section 1.3).

4. Cross reference the DLL load addresses with the shared memory segment addresses shown in the *onstat –g seg* output from step 1. If there is an overlap, and the DLL is a Microsoft system DLL, this is likely to be a Windows defect. If the overlap is caused by an IBM or third party vendor DLL, either report the problem to the vendor or manually rebase the DLL to another address where no memory segments are loaded using the **rebase** utility supplied with the Windows Platform SDK.

6.3.5. Solving Address Conflicts with SHMNOACCESS

Even with all Microsoft and IBM fixes, it is still possible to be in a situation where the oninit process creates a shared memory segment that overlaps with where another process that needs to attach to that segment has loaded a DLL, resulting in a "Unable to attach to shared memory" error.

The SHMNOACCESS onconfig parameter can be used to address this situation. Use it to exclude any regions of address space where you do not want oninit to create a shared memory segment. As of IDS v11.50.xC2, the SHMNOACCESS parameter works on all types of segments (resident, virtual, and so on). In versions between IDS v10.00.xC7 and IDS v11.50.xC1, the SHMNOACCESS parameter did not work for the resident segment and the first virtual segment.

As this parameter affects the layout of the shared memory, it takes effect when the instance is started. You can specify up to 10 different address ranges; as shown in the following example:

SHMNOACCESS 0x70000000-0x75000000, 0x7A000000-0x80000000

6.4 Security/Permissions problems

6.4.1 User Rights Restrictions at an Active Directory Server Level

Section 3.9.4 discussed the scenario of certain advanced user rights being restricted at an Active Directory Server Level. Typical restrictions that might be applied at this level include:

- Users not allowed to create other users
- User accounts not allowed to run as a Service
- User accounts not allowed to act as part of the operating system

Restrictions on running as a service will result in the IDS service not starting. Other user rights restrictions can cause apparently inexplicable -951 and -952 errors when users try to connect to IDS.

One way to view the effective informix user rights on a computer running Windows Server 2003, Windows Server 2008, or Windows Visa operating systems is to start a process as the informix user and run the following command:

whoami /all

The resulting list of user rights can be compared against the required user rights described in section 3.3.

6.4.2 Expired Passwords

By default the informix password is set to “password never expires”;, however, this default can be overwritten by the operating system or active directory security policies. Check also for installation failures due to non-compliance to the password policies of the system.

If a computer’s security policy requires user passwords to have a time limit and the informix user password expires, the first symptom might be that the IDS service fails to start.

A standard troubleshooting method for understanding service startup failures is to run the oninit process in the foreground as the informix user. In this instance starting a command shell as informix using *runas* or *ixsu* will fail due to the expired password.

When the informix password is changed, the **ixpasswd.exe** utility can be used to change the password for all Windows services that log on as the informix user (see Section 3.1.1).

6.5 Network/Connectivity Problems

6.5.1 Connection Attempts Time Out or Return -908 Errors

If a client connection to IDS appears to time out and no failed connection attempt is recorded in the online log there are several possibilities to explore. The possible causes are similar for all platforms:

- SQLHOSTS information on the client does not match the SQLHOSTS information on the server.

- A network configuration problem. The server machine cannot be pinged by the client or vice versa.

- A firewall issue. The server machine can be pinged but a telnet to the IDS listening port times out. Computers that have Windows operating systems installed are often unintentionally configured with more than one firewall. From Windows XP Service Pack 2 and Windows Server 2003 Service Pack 1, Windows has a default firewall, many network security products add an additional firewall, and routers that provide a subnet add another firewall. Every firewall protecting a computer on which IDS is installed needs to open the IDS TCP listening socket if remote client connections are required. See also section 3.5.

6.6. Troubleshooting Installation Problems

IDS installation problems can be categorized into two main areas: security and installing IDS on a computer that already has IDS installed.

Examples of security problems include submitting a password that does not match the complexity rules of the system, user rights not allowed by domain policies, or not having enough user rights to install the instance. For information about these problems refer to the information in Chapter 4 Security and to 6.4 Security/Permission Problems, above.

This section describes problems that might arise if you are installing on a computer where IDS was already installed. There might have been problems uninstalling the existing instance before installing the new instance. Perhaps the instance that you were trying to uninstall was still running and could not be stopped, or perhaps registry keys still exist.

If warnings are generating during installation, or the installation fails with error messages, one diagnostic method is to check whether a previous uninstall was effectively cleaned up from the computer.

Use the following steps when you are upgrading from a previous instance and are not trying to install into a new directory. In this case you would expect the uninstall program to delete INFORMIXDIR. However, you would expect the registry keys for the instance (HKLM\Software\Informix\Online\<instance name >) to still exist, as should the SQLHOSTS entries and the services entries. Those should only be deleted when you have selected “Remove server binaries and all database associated with them” during the uninstall.

To check the current status before you try to install again, the following diagnostic steps are suggested:

1. Does the INFORMIXDIR directory still exist?
2. If so, what files are still there?
3. Can these files be deleted? If not, what is accessing them?
   This can be checked with the Process Explorer.
4. Kill the program or service accessing the files and try to delete them.
5. Check the registry.

6. Which keys are still there? What is accessing them?

When you try to install into a new directory and want to keep the earlier installations, you would not want to delete INFORMIXDIR. In this case, the best method to find out what is going wrong during the installation is to monitor the installation with Process Monitor and to check for failures when reading keys and files, or where the installation stops.

Conclusion

The Windows port of IDS is a mature, native implementation that is well-integrated with the operating system. This document has detailed the technical details of this integration, and how to make use of this information for administration and troubleshooting.

One of the key advantages of Informix Dynamic Server is its ability to operate with minimal administration overhead.

We have attempted to capture the Windows platform-specific information that is required by an Informix DBA to save you time and administration costs.

We hope that you find the information herein to be useful and enjoy working with Informix Dynamic Server on Windows platforms.

We welcome and encourage your feedback, and any suggestions that you might have for improving the document.

References

The following manuals and Web pages were referred to while compiling this document and are suggested for additional reading.

Informix Resources

Informix Dynamic Server Information Center - complete online sets of IDS documentation –
http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp - IDS 10.0
http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp - IDS 11.10
http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp - IDS 11.50

IBM Informix Migration Guide -
http://publib.boulder.ibm.com/infocenter/idshelp/v115/topic/com.ibm.mig.doc/mig.htm

Performance Tuning Tips for IBM Informix Dynamic Server – IBM White Paper by Bobby Sukumar – a useful platform independent resource for IDS performance tuning -
http://www.ibm.com/software/sw-library/en_US/detail/F415342F55896R80.html

Security and Compliance Solutions for IBM Informix Dynamic Server - this is the IBM Redbook mentioned in Chapter 4 -
http://www.redbooks.ibm.com/abstracts/sg247556.html?Open

Customizing the Informix Dynamic Server for Your Environment one of a series of Redbooks dealing with IDS and relevant to the topics discussed here:
http://www.redbooks.ibm.com/abstracts/sg247522.html?Open

IIUG Software Repository Index –
http://iiug.org/software/software_index.html

Microsoft Network Technology

Inside I/O Completion Ports – Microsoft TechNet – an article explaining the I/O Completion Port technology used by the IDS KAIO and Network subsystems on Windows -
http://www.microsoft.com/technet/sysinternals/information/IoCompletionPorts.mspx

Networking and Access Technologies – Microsoft TechNet – a useful reference for Microsoft network technologies such as sockets, IPV4 and IPV6 -
https://www.microsoft.com/TechNet/network/default.mspx

TCP and NBT configuration parameters for Windows XP – a guide to tunable network parameters -
http://support.microsoft.com/default.aspx?scid=kb;EN-US;314053

Windows Performance

Tuning Windows Server 2003 on IBM xSeries Servers – an IBM RedPaper covering many aspects of Windows performance tuning, highly recommended –
http://www.redbooks.ibm.com/abstracts/redp3943.html

Windows Troubleshooting

Windows Sysinternals – the definitive source for Windows troubleshooting utilities and technical articles - http://www.microsoft.com/technet/sysinternals/default.mspx

Other Windows Operating System Resources

Naming Conventions in Active Directory for computers, domains, sites, and OUs – a source of reserved words and allowed characters for Windows user, machine and domain names – http://support.microsoft.com/default.aspx?scid=kb;en-us;909264&sd=rss&spid=3208

Windows Service Pack Roadmap – keep track of new service packs for specific versions of Windows - http://www.microsoft.com/windows/lifecycle/servicepacks.mspx

Windows Vista Developer Center – a collection of Windows Vista technical articles - http://msdn.microsoft.com/en-us/windows/default.aspx

Windows Server 2008 Developer Center - http://msdn.microsoft.com/en-us/windowsserver/default.aspx

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.