
insure/MONITOR

User Guide and Reference

Version 8.0

Document date: 09/15/2011



Centerfield Technology, Inc.

<http://www.centerfieldtechnology.com>

© 1999-2011 Centerfield Technology, Inc.

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of Centerfield Technology, Inc.

Notice: The information contained in this document is subject to change without notice. Centerfield Technology, Inc. shall not be liable for errors contained herein or consequential damages in connection with the furnishing, performance, or use of this material.

The sample source and resulting programs are provided for your use only. You may repackage any source or resulting program or portion of source or resulting program provided within this document for the purpose of product distribution, installation, or system archiving. Any other use requires the written permission of Centerfield Technology, Inc.

Centerfield Technology, Inc. and HomeRun are trademarks of Centerfield Technology, Inc.

All other product and brand names are trademarks and registered trademarks of their respective companies.

Use of this book and the information within implies an understanding and agreement with the above statements.

Centerfield Technology, Inc.

3131 Superior Drive NW - Suite C
Rochester, MN 55901



1	OVERVIEW OF HOMERUN	5
1.1	PRODUCT CONTENTS	6
1.1.1	<i>HomeRun installation</i>	6
1.1.2	<i>User Guide and Reference</i>	6
1.2	PRODUCT LIBRARIES	6
2	INTRODUCTION TO INSURE/MONITOR	7
3	GETTING STARTED WITH INSURE/MONITOR	8
3.1	HOMERUN REQUIREMENTS AND INSTALLATION	9
3.1.1	<i>Product Requirements</i>	9
3.1.2	<i>Product Installation</i>	9
3.1.3	<i>Connecting to the server for the first time</i>	18
3.2	USING ADDMON	21
3.3	SECURING THE USE OF HOMERUN TOOLS	23
3.4	CONFIGURATION FOR ONEWORLD® CUSTOMERS	25
3.5	INSURE/MONITOR VERSION 6.0 ENHANCEMENTS	29
3.6	INSURE/MONITOR VERSION 5.3 ADDITIONS	29
3.7	INSURE/MONITOR VERSION 5.2 ADDITIONS	30
3.8	INSURE/MONITOR VERSION 5.1 ADDITIONS	31
3.8.1	<i>JDE.INI editor</i>	31
3.8.2	<i>OneWorld® IFS log viewer</i>	33
3.9	INSURE/MONITOR VERSION 5.0 ADDITION	34
3.10	INSURE/MONITOR POLICY CONCEPTS	35
3.10.1	<i>User profile policy precedence</i>	35
3.10.2	<i>Time of day usage</i>	36
3.10.3	<i>Interfaces supported by insure/MONITOR</i>	36
3.11	USING INSURE/MONITOR TO AUDIT AND TRACK USAGE	41
3.11.1	<i>Issues</i>	41
3.11.2	<i>Challenges</i>	41
3.11.3	<i>Common Tasks</i>	42
3.11.4	<i>Reasons for auditing and tracking activity</i>	42
4	SCENARIOS FOR PRODUCT USE	46
4.1	USING INSURE/MONITOR TO SUPPORT USERS	46
4.1.1	<i>Introduction</i>	46
4.1.2	<i>Common Tasks with the User Monitor</i>	47
4.1.3	<i>Tips and techniques</i>	48
5	SUPPORT	51
5.1	CONTACT INFORMATION	51
5.2	ADDITIONAL INFORMATION	51
6	APPENDIXES	52
6.1	LOCK DETECTION AND DIAGNOSIS	52
6.2	SEMAPHORE WAIT DETECTOR	53
6.3	MUTEX WAIT DETECTOR	57
6.4	HISTORICAL DATA ARCHIVE (CFGMONHST)	59
6.4.1	<i>Overview</i>	59
6.4.2	<i>Configuration and setup</i>	59
6.4.3	<i>Archive reports and views</i>	62



6.5	CHECK COMMIT CONTROL (CHKCMTCTL).....	67
6.5.1	Overview.....	67
6.5.2	Configuration and setup.....	67
6.5.3	Examples for CHKCMTCTL.....	69
6.6	VISUAL EXPLAIN	70
6.7	USING VISUAL SQL EXPLAIN	71
6.7.1	Visual SQL Explain plan constructs.....	71
6.8	INTEGRATION OF ISERIES NAVIGATOR VISUAL EXPLAIN.....	83
6.9	SYSTEM-WIDE INSTALLATION IMPACTS.....	84
6.9.1	System Values	84
6.9.2	Servers, Jobs, and Monitors	85
6.9.3	Database Monitor and HomeRun.....	85
6.9.4	Subsystems and Work Management.....	85
6.9.5	Exit points.....	88
6.9.6	TCP/IP Usage.....	88
6.9.7	Job Scheduler Usage	89
6.9.8	Default public authority of libraries.....	89
6.9.9	Command Language (CL) Commands Used by HomeRun.....	89
6.9.10	OS/400 System APIs Used by HomeRun.....	90
6.9.11	Programs Adopting *OWNER Authority.....	91
6.10	ENABLING REMOTE MONITORING	94
6.10.1	Effects of exit point registration on your environment.....	94
6.10.2	Exit Program Registration Details	95
6.11	REMOVING REMOTE MONITORING SUPPORT	98

1 Overview of HomeRun

The HomeRun toolset allows you to efficiently design, deploy, manage, and support all aspects of an SQL-based environment. The set of tools in HomeRun includes:

- insure/INDEX
- insure/ANALYSIS
- insure/MONITOR
- insure/RESOURCES
- insure/SECURITY

All tools make use of the HomeRun server on the iSeries. The password you enter on the server controls which tools are available for your use. However, the manual for each of these tools is included in your product download.

In addition, the following tools are included with every HomeRun installation:

- Autonomic DBA
- Visual SQL Explain (a.k.a. sql/OPTIMIZER)
- Database Explorer
- Lock Detector
- Semaphore Wait Detector
- Mutex Wait Detector

See the Appendixes for more information on using these built-in tools.

1.1 Product Contents

The product you received from Centerfield Technology, Inc. contains the following items:

1.1.1 HomeRun installation

The download contains the software for both your iSeries and your Windows 2000/XP/Vista/7 client workstation. Refer to the Product Installation section for instructions on installing the software.

Also, for JDE OneWorld® (aka Oracle E1®) environments there is an additional piece of software that needs to be installed on the Windows® Terminal Server (i.e. Citrix® server).

1.1.2 User Guide and Reference

Manuals for all tools in the HomeRun toolset can be found in your product download. Each manual is intended to help you get started with the tool, explain the tool's features, and provide guidance on the effective use of the product.

1.2 Product libraries

Centerfield uses a naming convention where all IBM System i™ object names are prefixed with the letters “XC”.

As part of that naming convention throughout this manual the Centerfield *{program library}* will be “XCENTER80” and the Centerfield *{data library}* will be “XCENTERD80”.

This version of the product uses TCP/IP port number 9920 by default. The rest of this document refers to this port as the default port “**{default port}**”. If that port is in use, the installation will incrementally search for unused port numbers starting at **{default port}** and continue looking until one is found. To determine the port that was chosen use the following command:

```
WRKSRVTBLE SERVICE(CENTERFIELD_SERVER_80)
```

2 Introduction to insure/MONITOR

insure/MONITOR allows you to manage many different aspects of user activity in real-time. You can easily monitor end user activity and diagnose problems.

NOTE: To fully use insure/MONITOR support for remotely connected users (such as ODBC, DDM, or FTP users), you must enable remote monitoring with the use of the ADDMON command in the *{PROGRAM LIBRARY}* library. See the section ***Enabling remote monitoring*** for further information.

If you wish to view and manage all jobs on the system the use of the ADDMON command is optional, but all possible information will not be available for remotely connected jobs.

3 Getting started with insure/MONITOR

This section describes how to get started quickly with insure/MONITOR. It contains basic usage information for users who want to get started right away. For in-depth information using insure/MONITOR to support your users, see the next section.

The steps outlined in this Getting Started section are:

1. Installation of the product
2. Use of the ADDMON command to enable remote monitoring
3. Securing the product for authorized users only
4. **IF** using the OneWorld® application, follow the instructions in these document sections:
 - “Configuration for OneWorld® customers”
 - “Installation instructions for OneWorld® customers using Windows® Terminal Server (i.e. Citrix® server)”
5. Review of basic concepts used by insure/MONITOR
6. Begin using insure/MONITOR

3.1 HomeRun requirements and installation

3.1.1 Product Requirements

The HomeRun toolset requires specific System i and Windows hardware and software before it will install and perform correctly.

Here is the list of required IBM PTFs:

- ✓ V5R4:
 - MF39418, MF39419, MF39466, SI24100, SI23714, SI23713, SI24580, SI23891, SI23890, SI24569, SI24504, SI23485, SI25668, SI25041, SI23365, SI23514, SI28951, SI28952, SI28953, SI30076, SI30013, SI30014, SI31631, SI31633
- ✓ V5R4M5:
 - SI24100, SI23714, SI23713, SI24580, SI23891, SI23890, SI24569, SI24504, SI23485, SI25668, SI25041, SI23365, SI23514, SI28951, SI28952, SI28953, SI30076, SI30013, SI30014, SI31631, SI31633
- ✓ V6R1:
 - SI31727, SI31641, SI31407

NOTE: PTFs listed above are current with the release date of this document. Centerfield may learn of new PTFs after the document release so we strongly encourage you to check our website for latest PTF requirements – www.centerfieldtechnology.com

3.1.1.1 System i

- ✓ i5/OS V5R4, or V6R1, V7R1
- ✓ TCP/IP configured

3.1.1.2 Windows

- ✓ Windows 2000, Windows XP, Windows Vista, Windows 7
- ✓ Minimum 80486 @ 66Mhz, 32 MB RAM
- ✓ 50 MB of disk space for product
- ✓ TCP/IP installed and configured
- ✓ ODBC driver installed

3.1.1.3 Citrix Windows Terminal Server for E1

- ✓ Windows NT Server, Windows 2000 Server, Windows 2003 Server

3.1.2 Product Installation

The HomeRun toolset has software that needs to be installed on the System i and on the workstations. The installation processes for both the System i and the workstations are

designed to be simple interfaces that provide good default values. The following sections in combination with the on-screen documentation help you understand the installation processes.

If you're re-installing HomeRun you **MUST** refer to the “**HomeRun Installation Guide**” document rather than these 1st time installation instructions.

3.1.2.1 System i Installation

- 1) Before beginning the installation on the System i, please review this list of the impacts the installation will have on your system. For more detailed information about the HomeRun System i installation, see the Appendix titled *System-Wide Installation Impacts*.
 - ✓ This product uses TCP/IP to communicate between the System i and the personal computer. TCP/IP needs to be configured before the HomeRun server will function. Before continuing, make sure your Windows PC client can *ping* the System i system. See one of the following IBM System i references for information on configuring TCP/IP.
 - *TCP/IP Fastpath Setup (SC41-3430)*
 - *TCP/IP Configuration and Reference (SC41-3420)*
 - ✓ The installation uses TCP/IP port number **{default port}**. If that port is in use, the installation will search for unused port numbers starting at **{default port}** and continue looking until one is found. To determine the port that was chosen use the following command:

```
WRKSRVTBLE SERVICE(CENTERFIELD_SERVER_80)
```
 - ✓ The HomeRun server installs into a library named *{PROGRAM LIBRARY}*. *The installation process will copy over existing programs if the library exists.*
 - ✓ The HomeRun data files install into a library named *{DATA LIBRARY}*. *The installation process will selectively replace the data files and preserve information specific to your configuration and settings if they exist.*
 - ✓ The HomeRun server requires an active subsystem for proper installation and operation. Determine the subsystem you wish to use. It will be needed later in the installation process. If you have no special work management requirements, you may use the default of *{PROGRAM LIBRARY}/XCSBS80*.

-
- ✓ If you are installing over the top of an existing HomeRun or Database Essentials product library, the jobs currently using that library will be ended before the installation begins. The server will be re-started at the end of the installation if a valid password is entered during the install, or if you have installed over an existing library that already has a valid password.
 - 2) Sign-on to a 5250 session with QSECOFR or a user profile that has security officer special authorities. This user profile must have a minimum of *ALLOBJ, *IOSYSCFG, *JOBCTL, *SECADM, and *SAVSYS special authorities.
 - 3) The system value QUSEADPAUT must be set to *NONE.

You can verify and change this system value using the following command.

WRKSYSVAL QUSEADPAUT

In addition, if the system value QUSEADPAUT is secured with an authorization list, the user profile used for the installation must be on that authorization list.

- 4) Place the CD-ROM containing the HomeRun product into your System i primary CD-ROM drive and issue the following command (does not apply when performing a fully electronic installation).

LODRUN *OPT

- 5) When you are prompted to configure the HomeRun server subsystem, either take the default (i.e. XCSBS80) or enter the subsystem where you want the HomeRun server and client jobs to run, and press Enter. It is recommended that the default be taken. You can reconfigure the server's subsystem after the installation by running the CFGSVR command.
- 6) When you are prompted to enter the HomeRun license password, enter the license password that was provided with your product and press Enter. The license password is based on the serial number of the System i and the LPAR number. Hence, a different license password is needed for each System i that you install.

If you do not know the license password for your System i, you can skip this step of the installation by pressing F12. You can enter the license password after the installation by running the PASSWORD command from the {PROGRAM LIBRARY} library.

If you defer adding the license password, the HomeRun server will not start automatically. No client machines will be able to connect to the System i using any of the HomeRun tools until the password is entered.

-
- 7) You will be returned to an System i command line when the installation is complete. If you did not skip any of the installation steps, the HomeRun server will be automatically started for you. If you skipped any steps in the installation process, you should perform the configuration steps that you skipped before you continue.
 - 8) If you skipped configuring the server or the license password as part of the installation, or if the server did not automatically start as part of the product installation, you need to start the HomeRun server before clients can connect to the System i using any of the HomeRun tools. Prior to trying to start the server, you should ensure that TCP/IP is active on your system. If it is not active, issue the following command using the correct options for starting the TCP/IP servers on your system.

STRTCP STRSVR(*YES/*NO)

- 9) Issue the following command to start the HomeRun server:

{PROGRAM LIBRARY}/STRSVR

- 10) Verify the server is active in the subsystem that you specified. The job name will be XCSERVER. The following command can be used to check for the active job and to verify that the subsystem is correct.

WRKACTJOB JOB(XCSERVER*)

- 11) The server will need to be restarted every time the associated subsystem is ended or the system is restarted. You may add the *{PROGRAM LIBRARY}/STRSVR* command to your system startup routine (**recommended**) or add an autostart job entry in the subsystem if you would like to automatically start the server. If you start the server in your system startup procedure, the i5/OS command STRTCP must be issued and TCP must be completely started before the server will start.
- 12) If you plan to use insure/SECURITY or insure/MONITOR, you will need to register Centerfield's exit point programs with the System i registration facility:

{PROGRAM LIBRARY}/ADDMON

This enables the policies you define to take effect. More information can be found in the manuals for each of these tools.

NOTE: you will have to recycle the host servers for the exit point registration to take effect (refer to the table on next page as well as the Appendix for more detailed information).

-
- 13) If you plan to use insure/RESOURCES, you must install policy managers before your resource policies will take effect. This can be done either with the:

{PROGRAM LIBRARY}/ADDQCEXIT

or with the menus in insure/RESOURCES PC client GUI.

ODBC exit may need recycling database host server and corresponding prestart jobs (i.e. ENHHOSTSVR *DATABASE, ENDPJ QUSRWRK QZDASOINIT, STRHOSTSVR *DATABASE)

More information can be found in the insure/RESOURCES manual.

- 14) You are now ready to begin the workstation installation.

3.1.2.2 Windows 2000/XP/Vista/7 Installation

We recommend Administrator authority on the PC to run the install:

- 1) Go to website: <http://www.centerfieldtechnology.com/downloads.asp>
- 2) Click on the download link for HomeRun
- 3) Follow the online instructions in the START HERE pdf.
- 4) If the installation needed to replace DLLs, you will be prompted to restart your system.
- 5) You may now use the HomeRun tools for which you have a license.

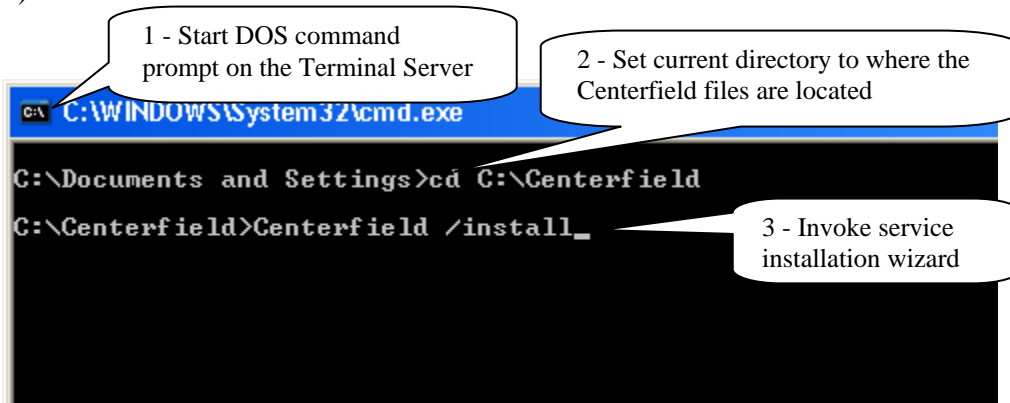
3.1.2.3 Installation instructions for OneWorld® customers using Windows® Terminal Server (i.e. Citrix® server)

These instructions apply **only** to customers that are running JDE OneWorld® (a.k.a. PeopleSoft EnterpriseOne® aka Oracle E1®) on their System i and have purchased insure/Monitor for OneWorld® license. Furthermore, customers using Terminal Servers (TS) in their environment and require Centerfield windows service running on those TS to achieve correlation of System i ODBC jobs to the OneWorld users will follow these instructions.

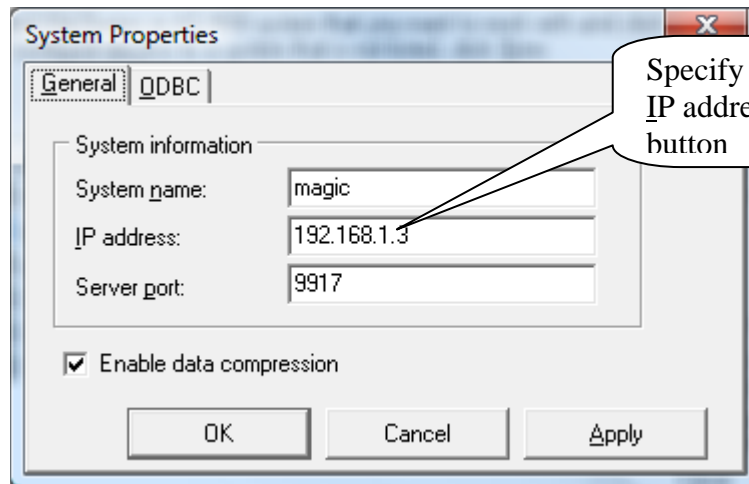
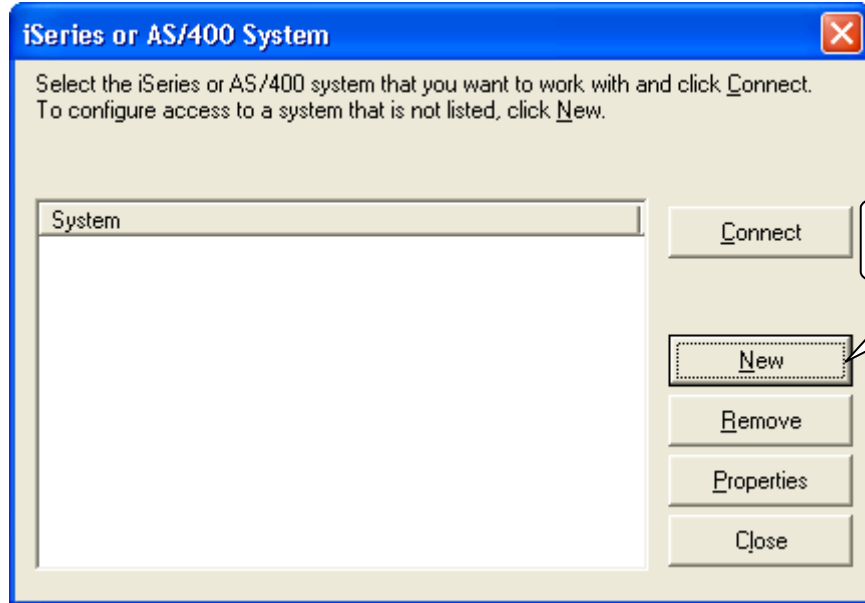
If these conditions do **NOT** apply to your scenario, do **NOT** follow these installation steps.

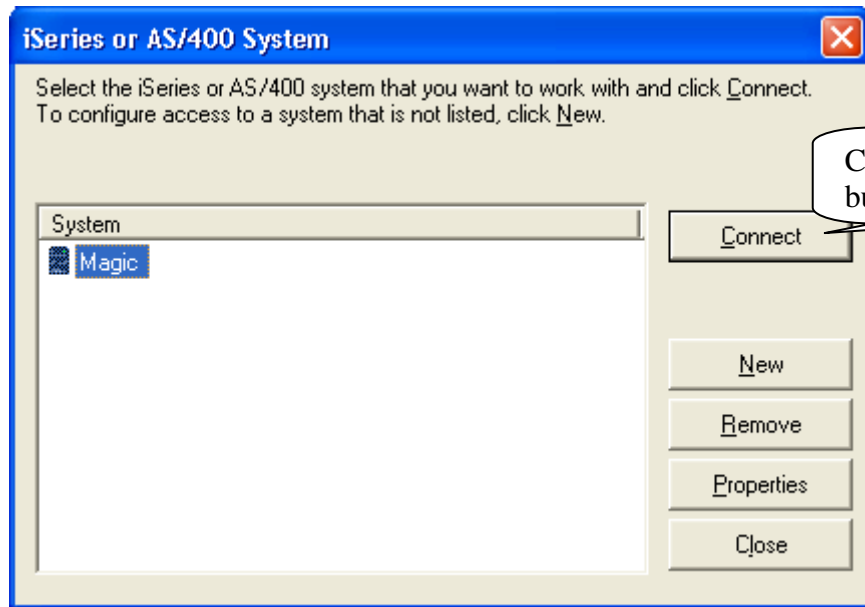
These instructions are to be performed by the Terminal Server administrator (i.e. OneWorld® CNC specialist) with Administrator authority to the server.

- 1) Locate **Centerfield.exe**, **Centerfield.dll** and **CenterfieldServiceConfiguration.exe** on the installation CD using Windows Explorer
- 2) Create a new folder on the Terminal Server (i.e. C:\Centerfield)
- 3) Copy and paste **Centerfield.exe**, **Centerfield.dll** and **CenterfieldServiceConfiguration.exe** to the newly created folder
- 4)

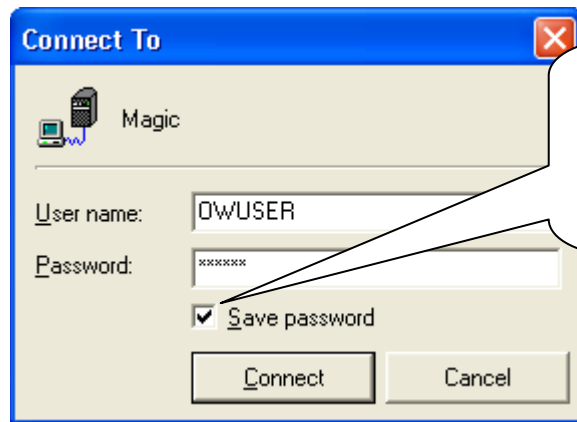


5) Several one time configuration windows will appear.

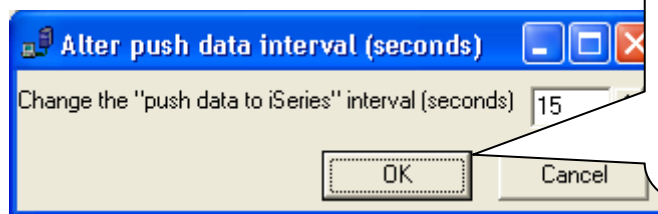




Click on Connect button



Specify iSeries profile and password, make sure you **check** the Save password checkbox then hit the Connect button.



Default 'push data to System i' interval is 15 seconds, but you can change it at this time. Good rule of thumb is to use 5 seconds per Terminal Server on which our service is running (i.e. $8 * 5 = 40$ seconds). When done click OK button



Upon successful installation of the Centerfield Terminal Server windows service you will see this window. Click OK to return to DOS prompt screen.

```
C:\WINDOWS\System32\cmd.exe

C:\Documents and Settings>cd C:\Centerfield
C:\Centerfield>Centerfield /install
C:\Centerfield>net start "Centerfield Service"
The Centerfield Service service is starting.
The Centerfield Service service was started successfully.

C:\Centerfield>_
```

start Centerfield window service

6)

You can check Centerfield service status in a couple of ways. One is to invoke the services applet via Start->Run->services.msc. Another is to check for the existence of Centerfield.exe process in the Task Manager on the Terminal Server.

If there are active OneWorld connections on the Terminal Server, there should be an XCCLIENT job on the System i with the joblog message:
“Serving Terminal Server at IP address 0.0.0.0.” where 0.0.0.0 will list the true IP address of the Terminal Server. If there are no active OneWorld connections on the Terminal Server, Centerfield service will not push any data to the System i.

This completes the Centerfield service installation on the Terminal Server. Service should auto-start every time Terminal Server reboots. For the Centerfield service to push data, XCSERVER job needs to be active on the System i. You can start it by executing *{PROGRAM LIBRARY}/STRSVR* command, or automate this by adding STRSVR command to your IPL startup routine (DPSYSVAL QSTRUPPGM).

There is one final configuration step that has to be performed from the insure/MONITOR for OneWorld® PC client. Refer to the documentation located under section **“Configuration for OneWorld® customers”** in the **“insure Monitor User Guide and Reference”** document for details.



3.1.3 Connecting to the server for the first time

Once you have the product installed, you are ready to connect to the server from the Windows based workstations. The workstation installation adds a program group to your start menu called *Centerfield HomeRun*. Within the program group there are one more features that can be selected depending on the options that were selected at installation time. To start the HomeRun tool you want to use, choose its name from this group.

The HomeRun tools use a consistent interface for connecting to the System i. When you start one of the HomeRun tools for the first time on your workstation, you will need to configure your connection to a server.

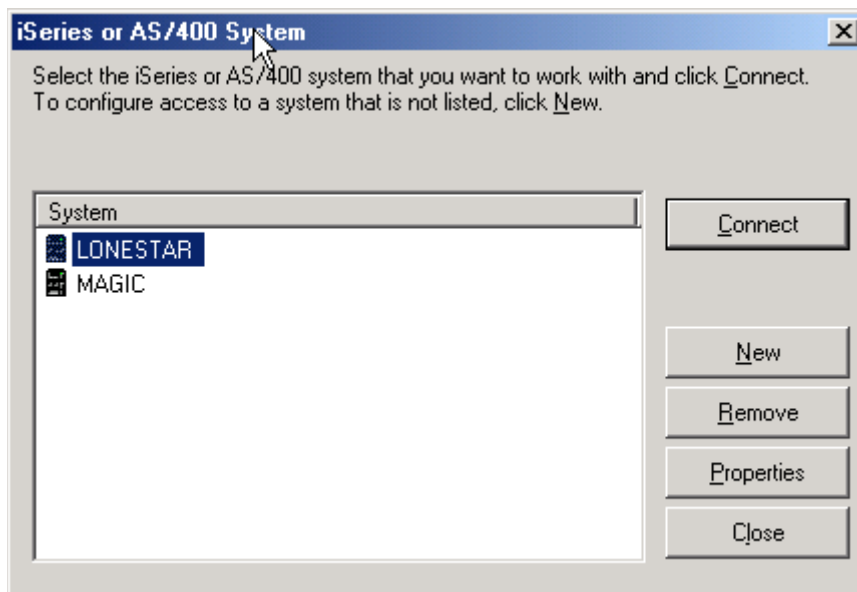
Auto-configuration

When you start the tool, it will try to auto-detect the System i systems in your environment and configure itself to connect to them.

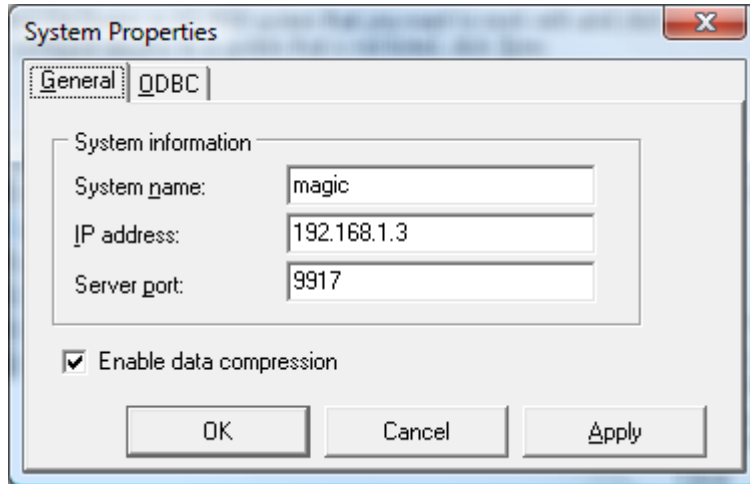
The auto-configuration interface will show you all of the systems that it can detect. If no systems are detected, the auto-configuration will be skipped and you will need to define your system connection information manually.

Manual configuration

If the auto-configuration interface did not detect your system, or you want to manually configure a connection, use the following instructions.



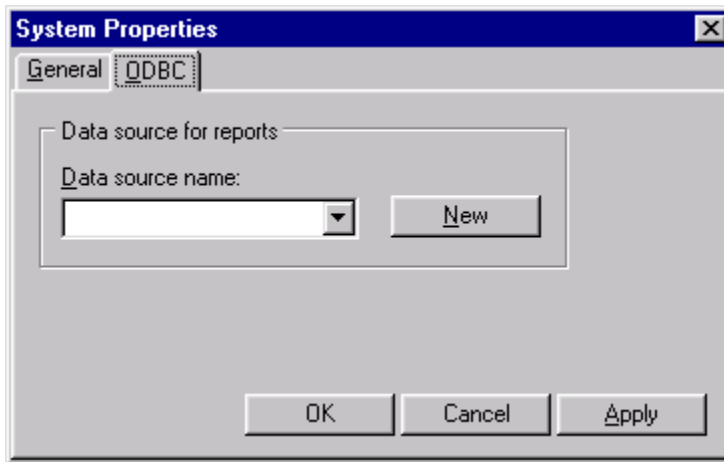
When the *System i system* window is displayed, press [New] to add a new system.



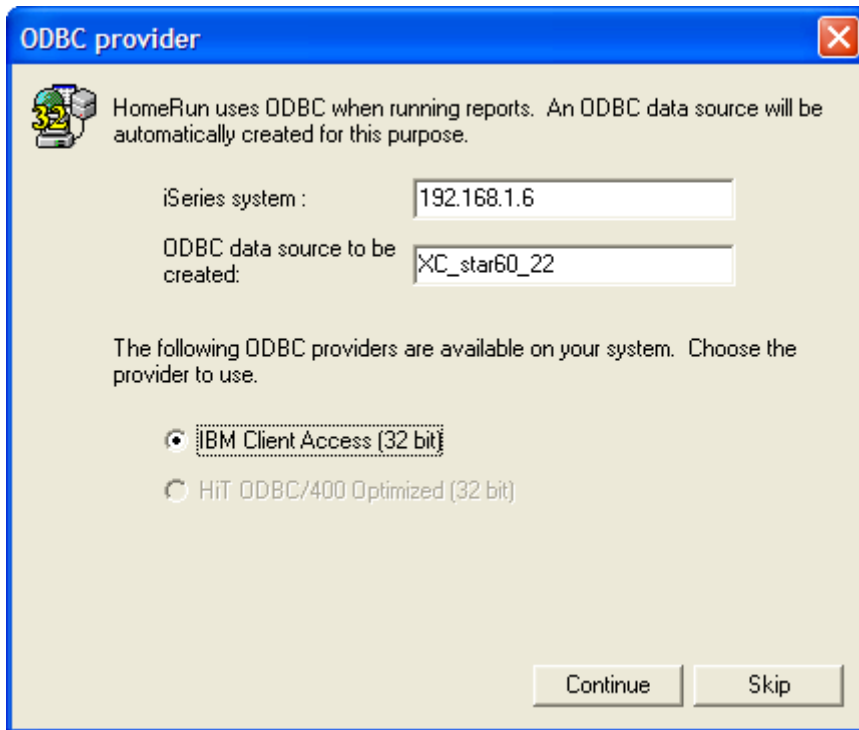
For *System name*, enter a descriptive name that identifies your system.

For *IP address*, enter either the symbolic name or dotted IP address for your System i system.

For the *TCP/IP port*, use the port that was selected during the System i installation process, by default **{default port}** is used.



On the *ODBC* tab, specify a name for your ODBC data source and press **[New]** to create a data source for this system. If you already have a Client Access ODBC data source that is configured to your system, you can specify your existing data source. If you choose to create a data source, you will see a display similar to the following.



Specify the name of the System i as defined in your connection software, and select the data source type that you want, and press **[Continue]**. You must have either Client Access or HIT ODBC installed to use this fast path interface for creating your data source. If you have another ODBC provider, you must create the data source with the ODBC Administrator utility that is provided with Windows and then specify the data source name within the *System Properties* window on the *ODBC* tab.

After you have created your data source and specified values for all of the appropriate prompts on the *System properties* window, press **[OK]** to create the connection definition.

Connecting to a system

Select the system that you want to work with from the list provided in the *System i system* window, and press the **[Connect]** button. Enter your user profile and password and press **[Connect]**.

3.2 Using ADDMON

Before policies you define in insure/MONITOR will take effect and to see all possible information for remotely attached jobs, you must run the Centerfield command ADDMON, found in the *{PROGRAM LIBRARY}* installation library.

This command tells the System i server which remote interface you are going to monitor by configuring the proper Centerfield programs at various IBM® exit points. You do not need to understand exit points to use this support; however, more detailed information is available in the Appendix entitled *Enabling Remote Monitoring*.

To enable insure/MONITOR to monitor all covered interfaces by default, specify *{PROGRAM LIBRARY}/ADDMON DRVTYPE(*ALL)*. This is the recommended default. If your environment requires that you pick and choose interfaces to which to add exit points, prompt on the ADDMON command and choose the interfaces.

The following table shows how the options on the ADDMON DRVTYPE parameter map to the options available for you to monitor in the insure/MONITOR interface:

ADDMON parameter	Interface Description
*CSCM	Central Server client manager
*CSCONV	Central Server conversion map
*CSLM	Central Server license manager
*DTAQ	Client Access - Data queue server
*DTAQORIG	Client Access - Original data queue server
*FILETRANS	Client Access - Original file transfer function
*FILSRV	File server

*IBMDDM	Distributed Data Management DRDA - FileTek products DRDA - Grandview DB/DC Systems products DRDA - IBM DB2 Connect (formally DDCS) DRDA - IBM DB2 for VSE and VM DRDA - IBM DB2 UDB for System i DRDA - IBM DB2 UDB for OS/390 DRDA - Informix Software products DRDA - Oracle Corporation products DRDA - StarQuest products DRDA - Wall Data Rumba for Database Access DRDA - XDB Systems products DRDA - Derby Network Client (DNC) DRDA - Java Client (JCC) DRDA - DataDirect Technologies (DDT) DRDA - SeeBeyond ICAN (Sun JCAPS)
*IBMFTP	IBM File Transfer Protocol (FTP) Client IBM File Transfer Protocol (FTP) Server IBM REXX File Transfer Protocol (FTP) Server IBM Trivial File Transfer Protocol (FTP) Server
*IBMODBC	IBM Client Access ODBC and Hit Optimized ODBC
*LMORIG	Client Access - Original License Manager server
*MSGORIG	Client Access - Original message server
*NETPRT	Client Access - Network print server - entry
*REXEC	REXEC server logon
*RMTCALL	Client Access - program call
*RMTCMD	Client Access - remote command
*RMTSQL	Client Access - Original Remote SQL server
*SOCKSRV	TCP signon server
*TELNET	Telnet
*VRTPRT	Client Access - Original virtual print server

The time at which a new exit program is recognized by the System i server varies from one exit point to another. Some take effect immediately; others, such as exit points on the Client Access host servers, do not take effect until the host server is next started. Still others, including the File Server, require that the subsystem (QSERVER in this case) be ended and restarted before the new program takes effect. If you find that insure/MONITOR is not having the effect you expected, end the server programs and restart them or, if possible in your environment, end the subsystem and restart it.

NOTE: IPL or bringing system to restricted state (ENDSBS *ALL) will refresh all server jobs so if you do any of the two actions regularly you can expect full exit point programs

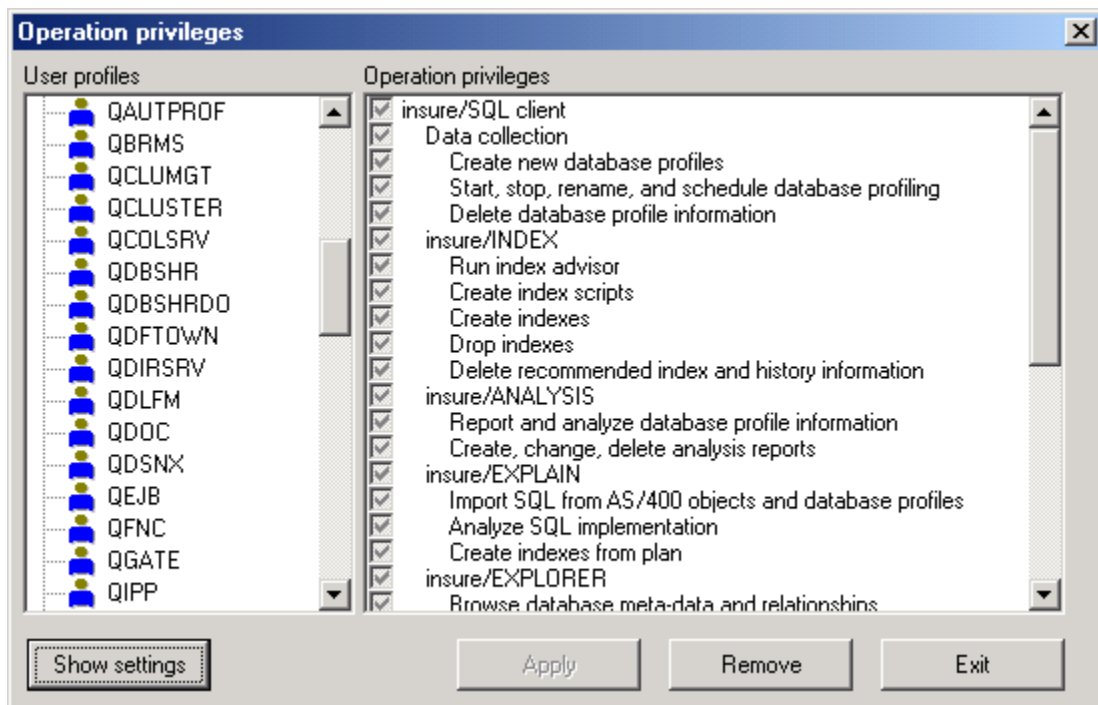
awareness and our product should behave normally.

3.3 Securing the use of HomeRun tools

HomeRun provides a broad suite of powerful functionality to the IT organization. Often, duties are split among various specialists. In this situation, it may be desirable to control what functions in the toolset each specialist can use. HomeRun provides a set of granular controls to allow an IT administrator to determine who can and should use each part of the toolset.

To implement security settings for a user or group profile use the following procedure:

1. In any of the HomeRun tools, select Tools | Privileges.
2. The next dialog has a list of users and groups on the left side and an itemized list of product features on the right as seen below:



3. To quickly restrict usage to a few users, select the *PUBLIC user profile and unclick the top-most box. At the same time, you should make sure one user profile has the capability to update privileges. To do that, pick one user profile and make sure *Administration* and *Edit HomeRun privileges* are checked. If you do not do this step, only the QSECOFR profile will still be able to edit privileges.

-
4. To change the settings for a particular user, click on their profile and push the “Show settings” button. On the right you will then see the parts of the product that profile is authorized to use. If the box is gray, the user does not have explicit authority defined and will get authority from either their group profile or global settings (the authority set for *PUBLIC).
 5. To restrict a user from a particular feature, simply uncheck the box in front of that feature and click the “Apply” button. If the user attempts to use that function, they will be given a message saying they are not authorized to that feature and that they should contact their system administrator.
 6. To remove settings, choose a profile (or set of profiles with multi-select) and click “Remove settings”. This will result in all settings being picked up from the group profile or *PUBLIC.

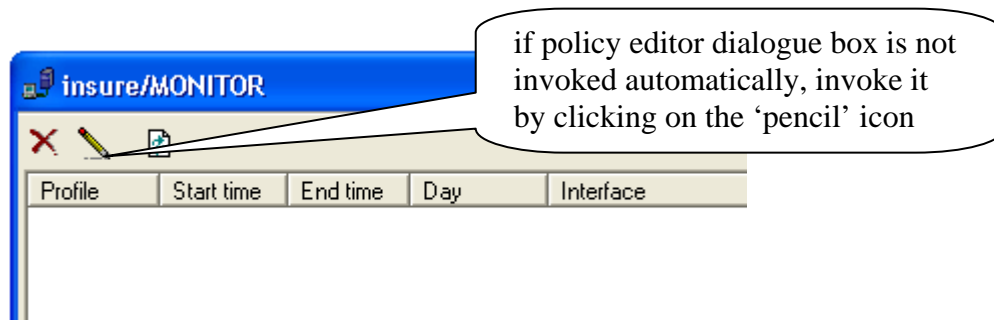
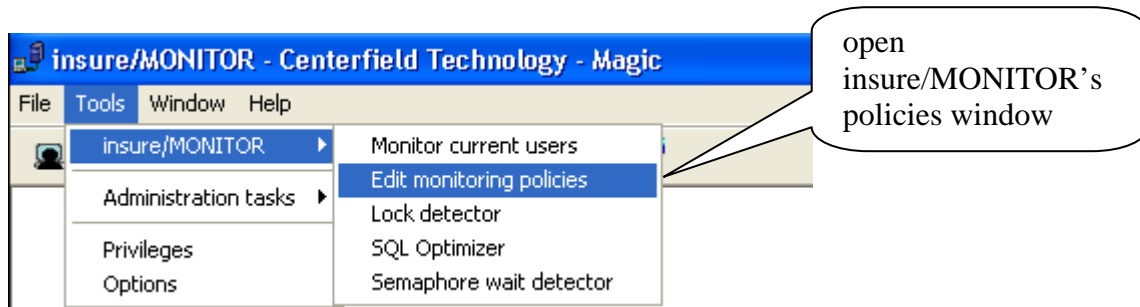
NOTE: When settings are changed for a user, they will not take effect until the user connects to that iSeries again (i.e. exits the client GUI and restarts it).

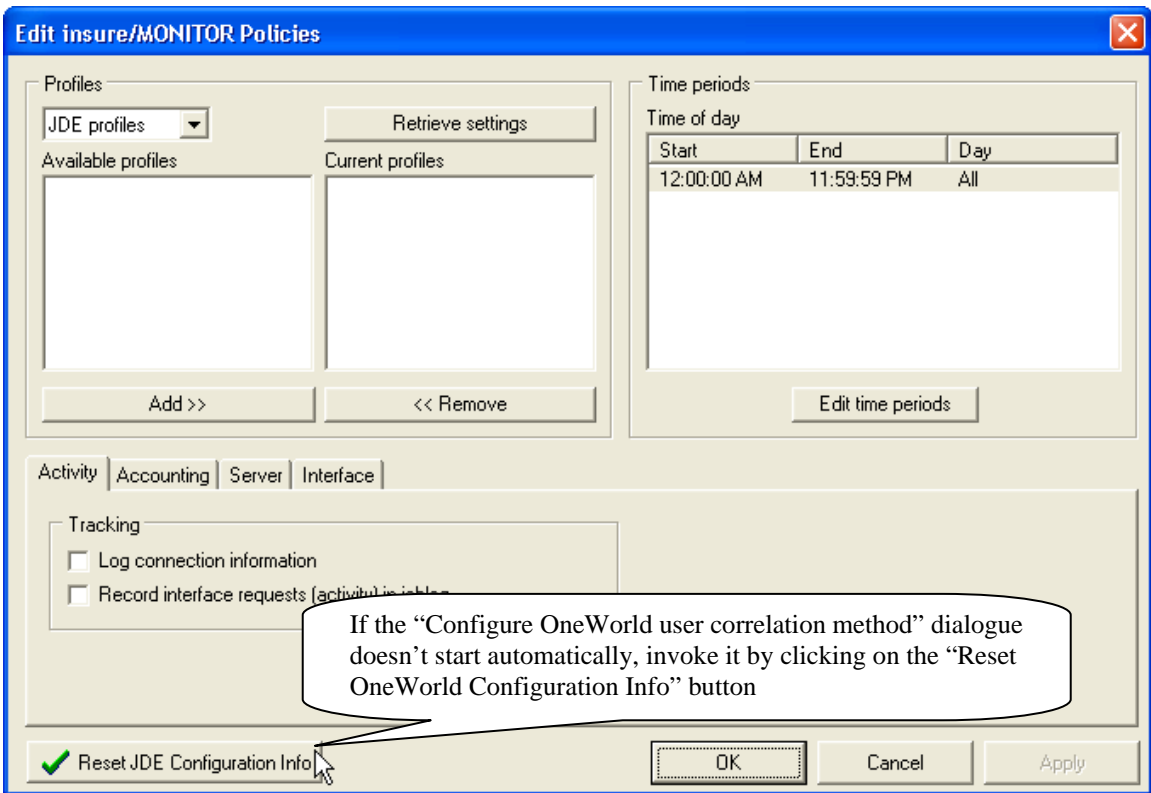
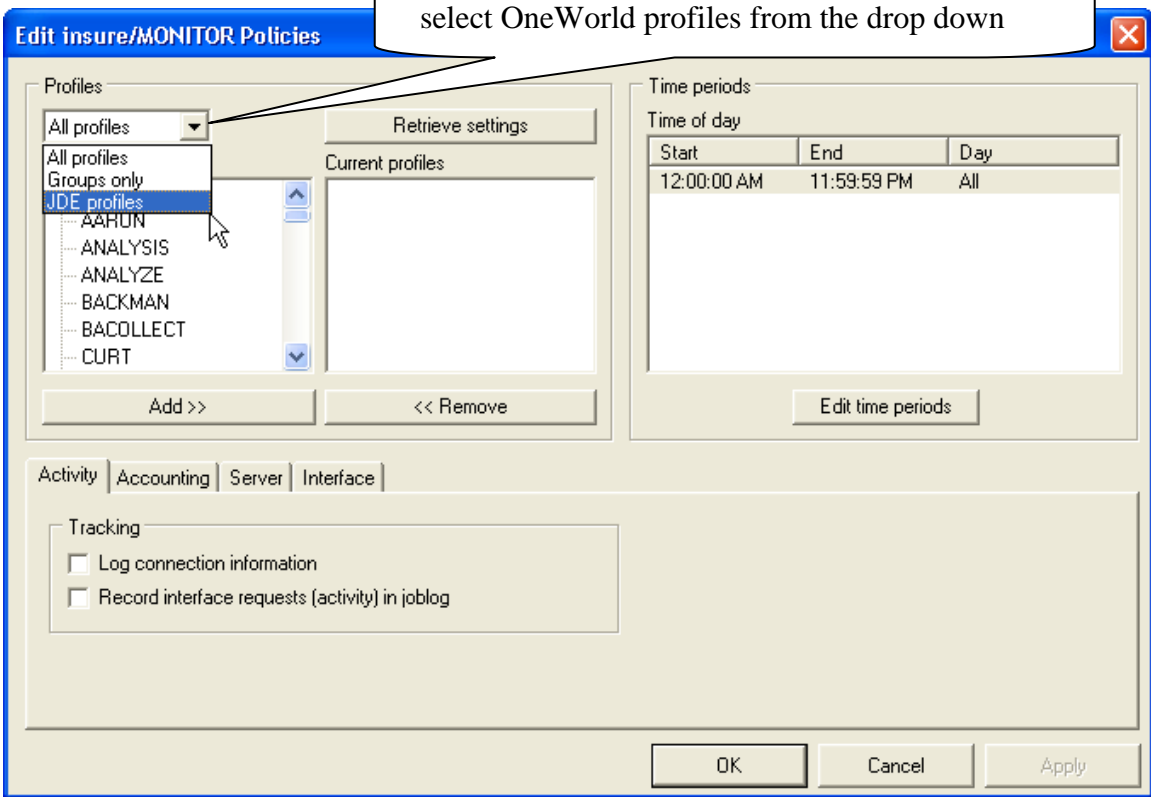
3.4 Configuration for OneWorld® customers

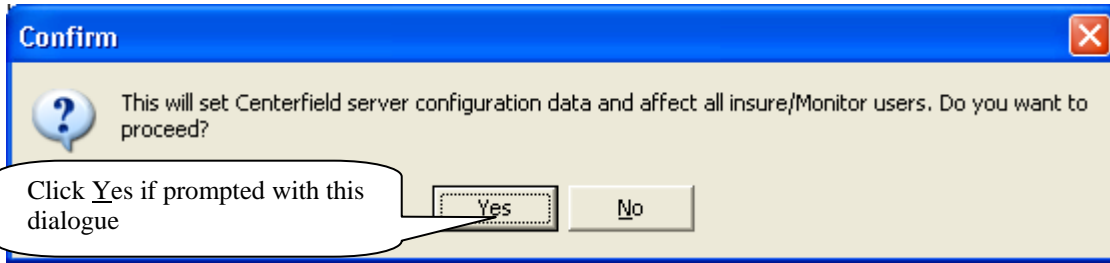
Do **NOT** execute these configuration steps unless you are OneWorld® shop and are using proxy profile(s) on the System i. These are **one-time, post-install** configuration steps and will take effect immediately.

To successfully correlate ODBC database jobs (QZDASOINIT) to the OneWorld® users that those jobs are doing work for, you need to install the Centerfield windows service on the appropriate Terminal Server (see “HomeRun requirements and installation” section). Additionally, you need to configure insure/MONITOR tool specifying the method of correlation and the location of the OneWorld® security table (i.e. SYS7333/F98OWSEC).

After starting the insure/MONITOR client (GUI) on your PC and connecting to the System i server, follow these steps:



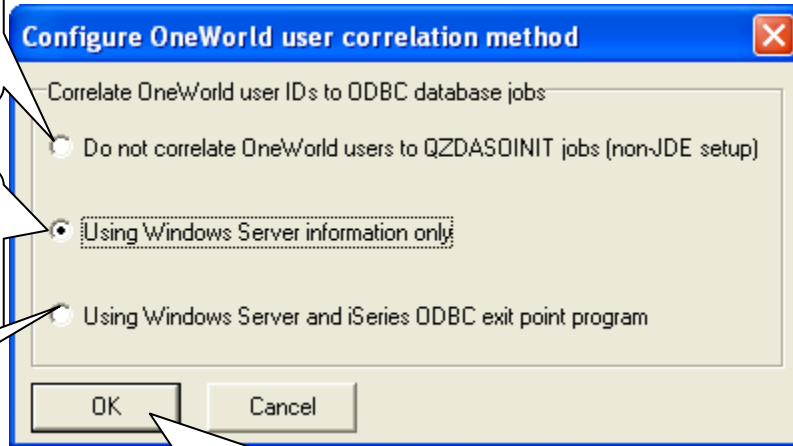




Select 1st option if your shop is not running OneWorld® ERP application or if you have created a matching System i user profile for each OneWorld® profile (no proxy System i profile)

Recommended option:
 Select 2nd option if OneWorld user name matches oexplore.exe windows process User Name (as seen in Task Manager dialogue on the Terminal Server)

Select 3rd option only if so instructed by the Centerfield support staff (uses dbmon to achieve OW user correlation)

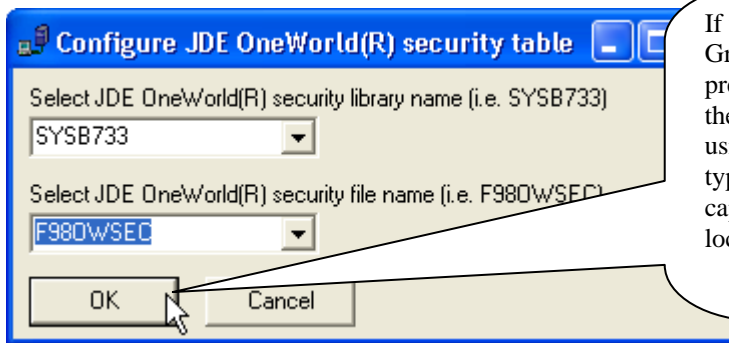
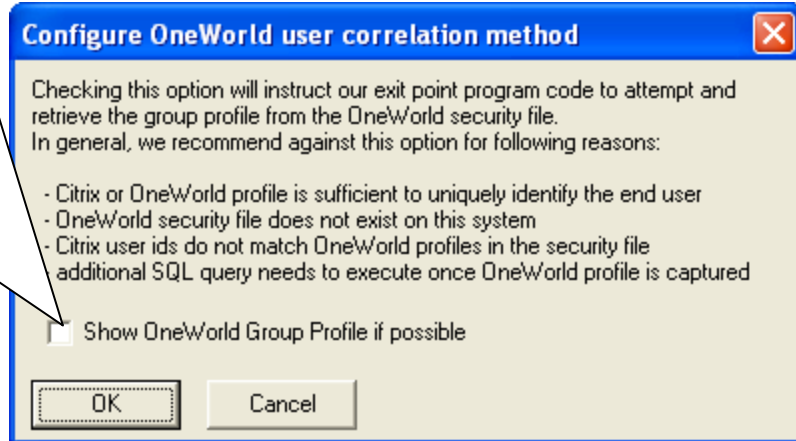


If you are unsure which option to select, contact Centerfield customer support for assistance. Once the appropriate option is selected, click the OK button.

Select this check-box only if:

- you want to see the OneWorld Group Profile populated in the insure/MONITOR column
- you have selected option 3 on the previous dialogue **OR** you have selected option 2 and the Citrix user profiles match OneWorld user profile (i.e. Unified Windows Logon in effect)
- OneWorld security file is located on this system (F980WSEC)

Centerfield recommends this option is left unchecked. Regardless if checked or not, you'll need to click OK to complete the configuration.



If you have checked the “Show OneWorld Group Profile if possible” checkbox on the previous dialogue, you need to set location of the OneWorld® security file on the System i using the drop down menus or by simply typing in the LIBRARY/FILE information in capital case. Once the correct security file location is set, click the OK button.

This configuration, in concert with the installation of the Centerfield windows service on the Terminal Servers (and any desired ‘fat’ clients), will allow insure/MONITOR to correlate the ODBC database jobs on the System i (QZDASOINIT) to the OneWorld user name (or Citrix user name) it is doing the work for.

This concludes configuration requirements for the insure/MONITOR environment. Provided that you have already registered IBM ODBC exit point program on the System i using our *{PROGRAM LIBRARY}/ADDMON* command (see “Using ADDMON” section for details) and installed Centerfield window service on the Terminal server (see “HomeRun requirements and installation” section for details), you can test the setup by starting insure/MONITOR GUI and selecting “OneWorld View” from the views drop down menu. You should see OneWorld® user and group ids correlated to the appropriate System i jobs (QZDASOINIT).

3.5 Insure/MONITOR version 6.0 enhancements

Version 6.0 of insure/MONITOR includes the following enhancements:

- Improved scalability for systems with a large number of jobs
- Support to view only jobs that have been active in the past 15 seconds
- Support to audit (log) job information for recently active jobs to minimize the amount of disk space and system resources to track information for inactive jobs
- Improved DNS reverse lookups to identify actual pc and server names
- The maximum size of an SQL statement that can be recorded in an iSeries joblog has been increased to improve problem diagnostics
- The OneWorld® Citrix service can now be configured while active
- OneWorld user names can now be logged in the joblog

3.6 Insure/MONITOR version 5.3 additions

With release of version 5.3, insure/MONITOR has seen some attractive additions.

Changes common to both versions:

- A mutex wait detection tool has been added. See Appendix for usage instructions. It has also been added to the Privileges function in case the insure/MONITOR administrator wants to disallow its usage to certain personnel.
- Added green-screen only CHKCMTCTL command to allow users to report and be alerted when commitment control boundaries have exceeded user defined boundary limits. This functionality is especially helpful to shops using save-while-active as the save will time-out and fail if it can't establish a synch-point due to extremely long active commit cycles. See Appendix for details.
- Added the ability to archive historical statistics seen in the insure/MONITOR GUI to a database file. Use the CFGMONHST green-screen command to turn this functionality on and set other configuration options. There are a number of sample Query Manager reports and SQL views shipped with the product. See Appendix for details.
- Made charts hot-clickable. Just right-click on the face of the bar in the bar graph or pie-slice in pie graph and select "Examine details for job" menu item. This will highlight the row in question and bring the jobs view into focus.
- Added 15 new charts
- Added 11 new statistical columns for the jobs
- Statistical delta points for "Since Last Refresh" columns for system jobs (new) as well as network (existed) are maintained and reported on. Check out CPU, Auxiliary I/O, Temporary Storage and Page Faults "Since Last Refresh" columns. This enhancement also makes CPU percentage column valid and accurate for system jobs, but does require 2 quick consecutive refreshes to ensure percentage formula time denominator does not "water down" the percentage and make it

-
- converge to zero percents.
 - Scaling performance enhancements
 - Internationalization code for running under non-English CCSIDs
 - Expanded and improved client connection IP address capturing.

Changes unique to insure/MONITOR for OneWorld® users:

- Windows service running on Terminal Server (i.e. Citrix) and/or has been solidified. Detailed logging added.
- Group profile requirement for the OneWorld configuration removed.
- Recommended configuration option is now Citrix user correlation, to reduce the overhead of the previous correlation method where exit points were used to start the database monitor.
- Added new jobs filter for OneWorld system jobs and all ODBC jobs, not just the ones we captured the Oneworld (Citrix) user profile for.
- A new Privileges function now allows insure/MONITOR administrator to disallow access to the Edit JDE.INI function.

3.7 Insure/MONITOR version 5.2 additions

With release of version 5.2, insure/MONITOR:

- A new semaphore detection tool has been added. See Appendix for usage instructions
- Page Fault chart has been added. Click on Charts to view it
- Visual Explain tool has been surfaced to the Welcome screen

Enhancements for OneWorld® users:

- Code running on Terminal Server (i.e. Citrix) and/or ‘fat’ clients has been rewritten as Centerfield windows service. Installation is simple, one-time process (see “HomeRun requirements and installation”). The administrator no longer needs to be logged on the Terminal Server (TS) for the service to perform its work.
- Additional error logging to Windows Event Viewer for the Centerfield Service running on Terminal Server
- Alternative and better performing OneWorld User-to-ODBC-job correlation method has been established, so administrator can now select between purely TS correlation method or a method using combination of TS and System i ODBC exit point code.

The first method should be preferred if the OneWorld user name matches the windows user name on the TS, as seen in the Task Manager processes tab for the oexplore.exe process.

For more detailed description on which option to select, contact Centerfield customer support.

3.8 insure/MONITOR version 5.1 additions

Additions include:


- Integrated Centerfield Technology Visual Explain
- Integrated iSeries Navigator Visual Explain

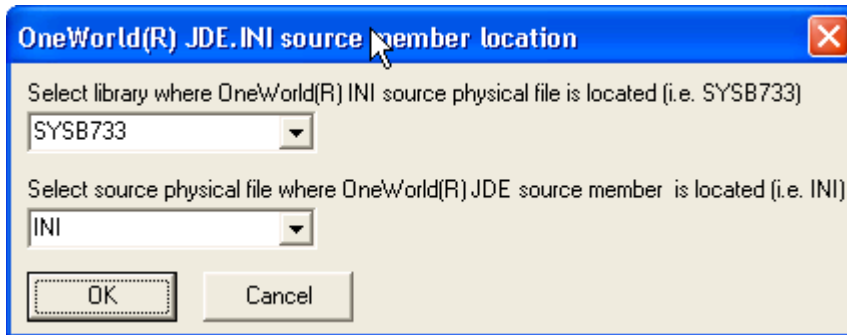
Changes specific to OneWorld® customers:

- Added ability to display OneWorld® users and group for any active and inactive UBE jobs (JOBQ status).
- Added System i JDE.INI configuration file editor
- Added OneWorld® IFS logs viewer for both regular and debug log files.

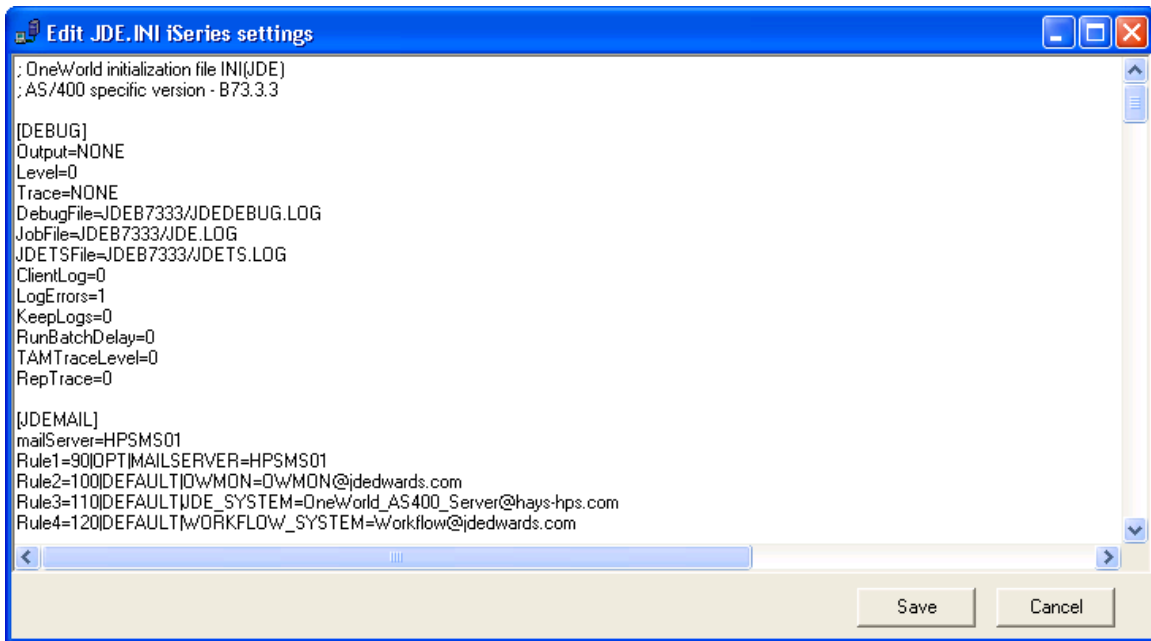
3.8.1 JDE.INI editor

The JDE.INI editor makes editing of initialization settings for OneWorld® easy when the need arises to edit these values. The typical scenario is that the DEBUG option must be set to get debug information in a text file, which you can later view from within insure/MONITOR with our JDE IFS log viewer.

You would simply click on the  icon to invoke the editor and specify where the JDE.INI is located on the System i (one time configuration) using following dialog box:

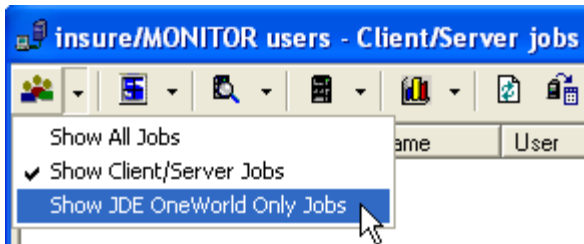


Once the data is retrieved from the System i you are free to edit it. To enable debug you would need to set the Output line to something other than NONE. Here is a screen shot of the editor window:

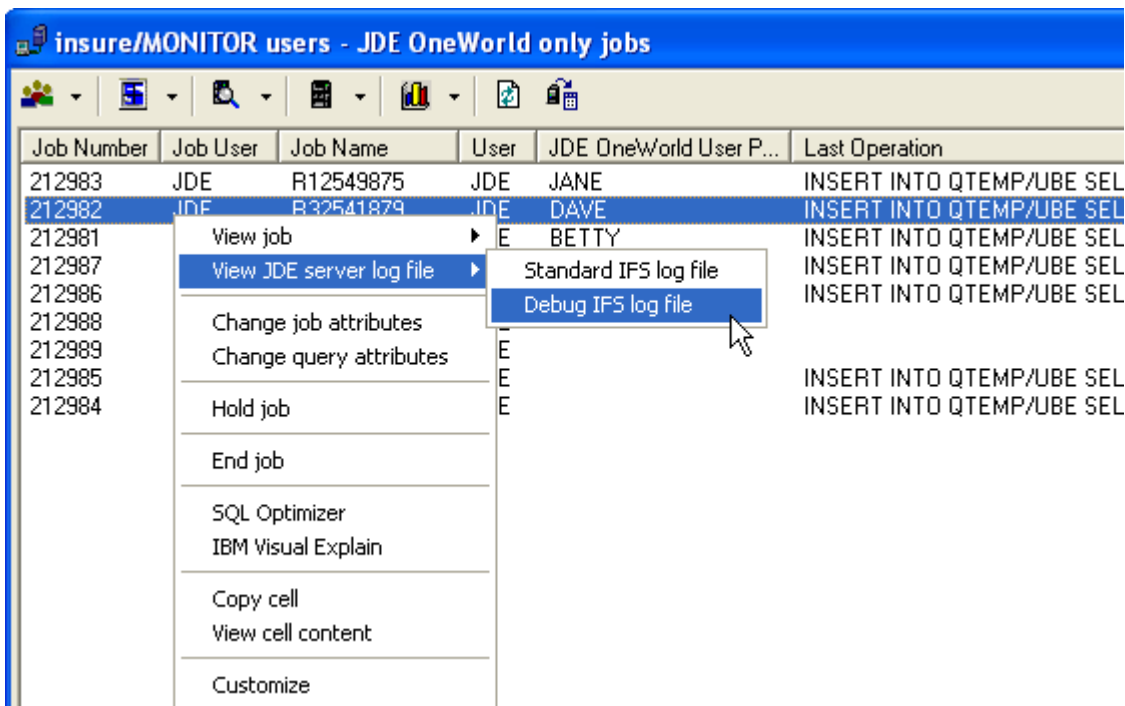


3.8.2 OneWorld® IFS log viewer

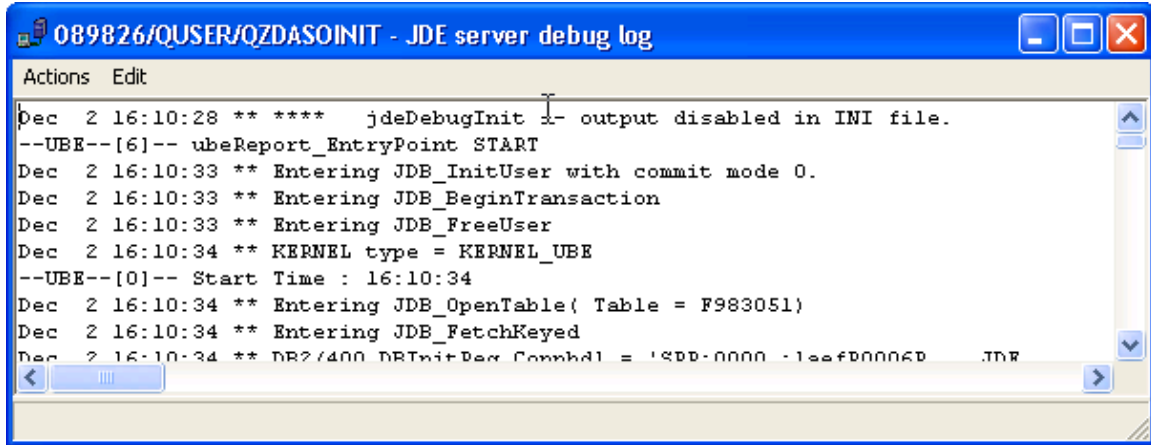
You can view the OneWorld® IFS log files in the built-in log viewer. First, make sure you switch from the regular “Client/Server Jobs” view to the “OneWorld Only Jobs” view. This view will include any ODBC connections we’ve captured the OneWorld user for as well as any other system jobs we deem belongs to OneWorld® based on the job naming convention OneWorld uses (kernel, sentinel, UBEs).



Next, select a UBE job you want to view the log for, right-click and pick either Standard IFS log file or Debug IFS log file.



This is a sample of the viewer display:



```
089826/QUSER/QZDASOINIT - JDE server debug log
Actions Edit
Dec 2 16:10:28 ** ****   jdeDebugInit  output disabled in INI file.
--UBE--[6]-- ubeReport_EntryPoint START
Dec 2 16:10:33 ** Entering JDB_InitUser with commit mode 0.
Dec 2 16:10:33 ** Entering JDB_BeginTransaction
Dec 2 16:10:33 ** Entering JDB_FreeUser
Dec 2 16:10:34 ** KERNEL type = KERNEL_UBE
--UBE--[0]-- Start Time : 16:10:34
Dec 2 16:10:34 ** Entering JDB_OpenTable( Table = F983051)
Dec 2 16:10:34 ** Entering JDB_FetchKeyed
Dec 2 16:10:34 ** DB2/400 DBInitDev Connhd1 = 'SDB-0000 -1ee4D0006B JDE
```

The Log viewer provides search, select, copy, print, font and other formatting options.

3.9 insure/MONITOR version 5.0 addition

With release of version 5.0, insure/MONITOR has new extensions to support OneWorld® environments.

Special functionality has been added to accommodate the OneWorld® environment including:

- 1) The ability to correlate users OneWorld® ODBC connections with System i jobs (QZDASOINIT) jobs.
- 2) Unique view displaying only OneWorld® ODBC jobs as well system kernel tasks, UBEs and SENTINEL jobs.
- 3) Unique graph displaying a summary distribution of usage of critical system resources by OneWorld® job as opposed to the rest of the system workload.
- 4) Ability to define diagnostic policies for OneWorld® users and groups.

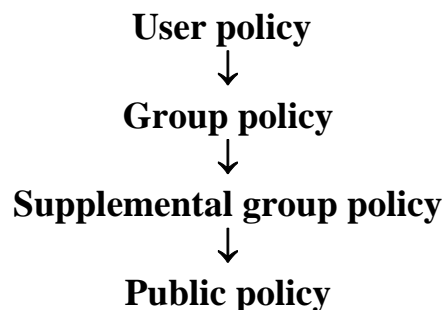
3.10 insure/MONITOR policy concepts

Central to the functions in insure/MONITOR is the concept of a policy. A policy defines a set of rules about when to apply certain restrictions, which users are affected by the rules, and the interfaces into the system for which the policy applies. Before discussing the specific details about each policy editor the following topics will be covered:

- Use of user profiles
- Time of day definitions
- Policy use with given interfaces

3.10.1 User profile policy precedence

insure/MONITOR can define policies for a user profile, a group profile, supplemental group, or for everyone on the system. If a policy is defined for everyone on the system (*PUBLIC) then that rule will be applied to all users unless they have a rule which applies to their specific profile or group profile. The search for a policy that applies to a given user is identical to the one used for System i security:



For example, if you define a rule for *PUBLIC which says all queries must run in less than 20 minutes, all users will have that rule applied when they connect to the system. If you have a group profile defined for everyone in the accounting department, you can define another rule that allows queries for that group to run for 40 minutes. Since a group profile exists for people in accounting, the rule for *PUBLIC will be ignored. You may also define a rule for your CFO that allows him to run queries of any length. Since a rule for a specific profile (the CFO) has been defined, both the accounting group and the *PUBLIC rules will be overridden.

Tip In general it is best to use *PUBLIC or group profiles to minimize the work required to manage users. If you can define user “classes” and group people into those classes, it reduces the number of policies you are required to define and maintain.

3.10.2 Time of day usage

Policies defined with insure/MONITOR are based on a time of day. By default, all rules apply for all hours of the day but you may want to use different policies for different times. For example, if your System i is not used heavily after 10:00pm, you may give users the capability to use more system resource by either running their jobs at a higher priority or allowing the use of queries that use more than one processor.

3.10.3 Interfaces supported by insure/MONITOR

Each policy you define can be applied to a wide variety of interfaces, or access methods. This allows you to easily customize policies based on access method or even control which access methods you support. Once a rule has been defined, you identify which interfaces to apply the rule to by simply checking a set of boxes.

Supported interfaces include those described in the following four tables:

Client Access Original Servers

The original servers were used by Client Access for DOS, Client Access for DOS with Extended Memory, and Client Access for OS/2 Functions. You can monitor or secure these exit points to ensure any clients using these host servers conform to your environment.

Note: HomeRun implements its monitoring and security for these servers via exit points registered in the registration facility. To have the system check the registration facility for these programs, you must set the PCSACC network attribute to *REGFAC. See IBM's documentation for the Change Network Attributes (CHGNETA) command for help if you need to change this attribute.

insure/RESOURCES and insure/SECURITY name	Description	Enforced when ADDMON parameter is used:
Client Access – Original virtual print server	Used by original Client Access clients' Virtual Print function. Allows use of a printer that is attached to the host system as though the printer was directly attached to a personal computer.	*VRTPR, plus PCSACC = *REGFAC
Client Access – Original data queue server	Original Client Access provides APIs that use this server to allow PC applications to work with System i data queues.	*DTAQORIG, plus PCSACC = *REGFAC

Client Access – Original file transfer function	PC Support File Transfer Function and HIT ODBC for file transfer. It also refers to Client Access File Transfer Function for Version 3 Release 1 Modification 3 or early versions.	*FILETRANS, plus PCSACC = *REGFAC
Client Access – Original message server	Original message function routes messages that are sent from PC users to the appropriate user and receives messages for PC users and sends them to the PC workstation.	*LMORIG, plus PCSACC = *REGFAC
Client Access – Original License Manager server	Original license server ensured valid licenses for original Client Access clients.	*MSGORIG, plus PCSACC = *REGFAC
Client Access – Original Remote SQL server	Original Client Access provides APIs that use this server to allow PC applications to run SQL statements on an System i.	*RMTSQL, plus PCSACC = *REGFAC

Client Access Host Servers

These servers are used by Client Access Express, and may be used by other products. These are the servers which are started by the STRHOSTSVR command.

insure/RESOURCES and insure/SECURITY name	Description	Enforced when ADDMON parameter is used:
Central Server client manager	Runs an exit program for all client management requests received by the central server.	*CSCM
Central Server conversion map	Retrieves conversion maps for PC applications which require them when they connect to the System i.	*CSCONV

Central Server license manager	Used for license management requests, including those from Client Access.	*CSLM
Client Access – program call	Allows client applications to call System i programs and pass parameters.	*RMTCALL
Client Access – remote command	Allows client users and applications to issue System i CL commands.	*RMTCMD
Client Access – Data queue server	APIs that can allow PC applications to work with System i data queues.	*DTAQ
Client Access – Network print server – entry	Allows enhanced client control over print resources on the System i server.	*NETPRT
File server	Allows clients to store and access information, such as files and programs, located on the System i server.	*FILESRV
TCP signon server	Runs when various signon requests are received, including Retrieve signon information, Change password , and Generate authentication token	*SOCKSRV

Other Registered Exit Points

These interfaces are monitored and secured by HomeRun via exit programs registered with the registration facility.

insure/RESOURCES and insure/SECURITY name	Description	Enforced when ADDMON parameter is used:
IBM File Transfer Protocol (FTP) Client	Controls access to the System i FTP client at validation time.	*IBMFTP
IBM File Transfer Protocol (FTP) Server	Controls access to the FTP server at validation time.	*IBMFTP
IBM REXEC Server	Controls access to Remote Execution Server at validation time.	*IBMFTP



IBM Trivial File Transfer Protocol (TFTP) Server	Trivial file transfer protocol (TFTP) provides basic file transfer with no user authentication. This protocol provides support for the IBM Network Station for System i.	*IBMFTP
IBM Client Access ODBC and Hit Optimized ODBC	The Client Access ODBC driver, HIT ODBC based on the Optimized Database Server, Client Access File Transfer Function, and other interfaces that use the Optimized Database Server.	*IBMODBC
REXEC server logon	Controls the authentication of users to a TCP/IP application server at logon time.	*REXEC
Telnet	Hooks into the Telnet interface's signon logic.	*TELNET
Workstation gateway server sign-on	The server which transforms 5250 data streams to HTML for dynamic display on web browsers.	*WSG

DDM Server and Other interfaces

These interfaces are monitored and secured by HomeRun via the DDMACC network attribute.

insure/RESOURCES and insure/SECURITY name	Description	Enforced when ADDMON parameter is used:
Distributed Data Management	IBM DDM	*IBMDDM
DRDA – FileTek products	FileTek products which use the DRDA interface	*IBMDDM
DRDA – Grandview DB/DC Systems products	Grandview products which use the DRDA interface	*IBMDDM
DRDA – Informix Software products	Informix products which use the DRDA interface	*IBMDDM



DRDA – IBM DB2 for VSE and VM	Connections from IBM DB2 for VSE and VM which use the DRDA interface	*IBMDDM
DRDA – IBM DB2 Connect (formerly DDCS)	Connections from DB2 Connect	*IBMDDM
DRDA – IBM DB2 UDB for System i	Connections from IBM DB2 UDB for System i which use the DRDA interface	*IBMDDM
DRDA – IBM DB2 UDB for OS/390	Connections from IBM DB2 UDB for OS/390 which use the DRDA interface	*IBMDDM
DRDA – Oracle Corporation products	Oracle products which use the DRDA interface	*IBMDDM
DRDA – StarQuest products	StarQuest’s ODBC and DRDA products including StartSQL, StarPipes, and Host Data Replicator. If you are using the ODBC driver that is shipped as part of Windows BackOffice SNA server or other Microsoft product, StarQuest is most likely the manufacturer.	*IBMDDM
DRDA – Wall Data Rumba for Database Access	Wall Data products which use the DRDA interface	*IBMDDM
DRDA – XDB Systems products	XDB Systems products which use the DRDA interface	*IBMDDM
DRDA - Derby Network Client (DNC)	Derby Network products which use the DRDA interface	*IBMDDM
DRDA - Java Client (JCC)	Java Client software which use the DRDA interface	*IBMDDM
DRDA - DataDirect Technologies (DDT)	DataDirect Technologies products which use the DRDA interface	*IBMDDM

DRDA - SeeBeyond ICAN (Sun JCAPS)	SeeBeyond products which use the DRDA interface	*IBMDDM
--	---	---------

3.11 Using insure/MONITOR to audit and track usage

This section introduces the use of insure/MONITOR. It describes the purposes for which insure/MONITOR can be used, and explains how to best use the features of the product.

3.11.1 Issues

To effectively manage any environment, it is essential that the administrator understand the “who, what, where, and when” information about activity occurring on their system. Audit information arms them with the necessary facts to set policies, change system configuration, predict workload, and tune the System i or database.

A lot of information about *general* system health can be collected from existing products such as the IBM Performance Tools or the built-in system auditing feature. In today’s environment however, more information is needed if the administrator is to have enough detail to make decisions based on real-world data. For example, in a data warehouse environment it is important to understand what data is not being used in order to make decisions about controlling the growth of the warehouse. This information is difficult (and in some cases impossible) to obtain using traditional systems management tools.

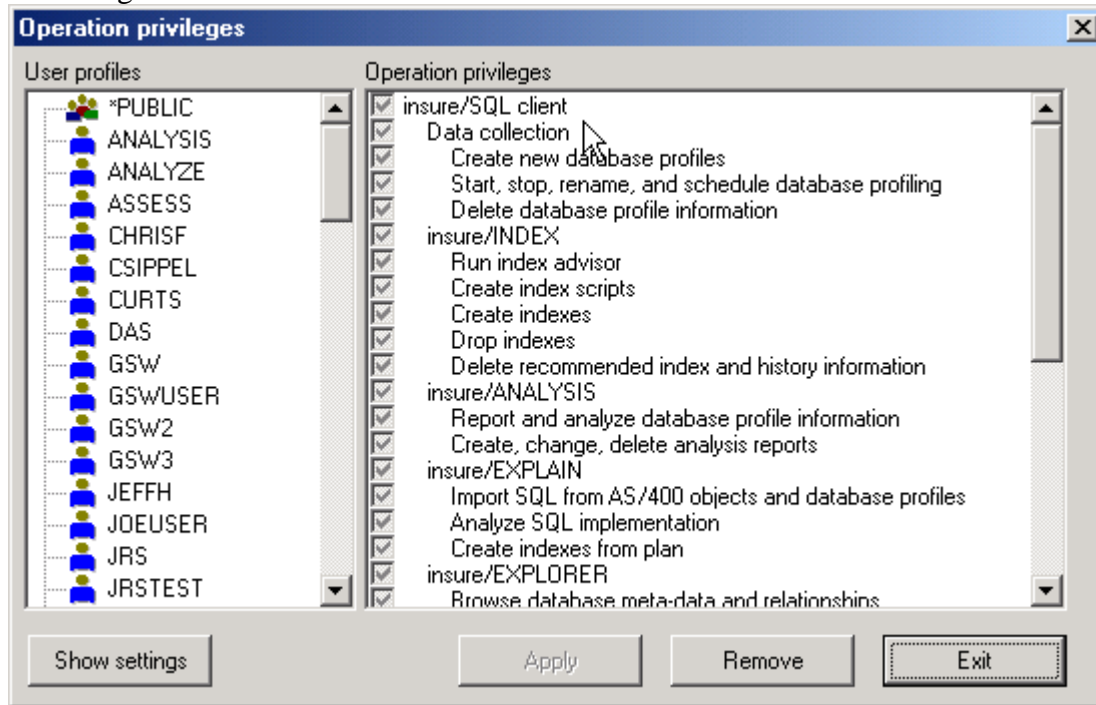
3.11.2 Challenges

The challenge with these environments is to provide system administrators the necessary information for them to do their jobs in a proactive manner. insure/MONITOR provides the capability to capture data in a flexible, easy-to-use way.



3.11.3 Common Tasks

The following sections talk about several different tasks that can be performed using insure/MONITOR. The main window for policy definition looks similar to the following:



You can define policies based on a user, group of users, or *PUBLIC. The policies can also be scoped to a particular time of day and specific interface or set of interfaces. This allows a great amount of flexibility and the ability to change policies based on their intent.

3.11.4 Reasons for auditing and tracking activity

There are many different reasons to audit and track activity. Depending on the reason, you will define different policies to fit that purpose. The following paragraphs describe the possible reasons to audit and track activity and their implications.

Help desk diagnostics: A system administrator may want to easily diagnose problems reported by end-users. To do that an administrator needs information on what the end-user is doing and potentially other information collected about a sequence of operations or error messages. The following settings are best suited for this type of use:

- Joblog information
 - Record interface requests in the joblog
 - Create a joblog for each connection
 - Put the joblog in a pre-defined System i output queue so it can be quickly

located

Recommended settings:

- Due to the overhead of collecting this information it is recommended that these settings be turned on when they are needed to diagnose a problem.
- Since putting a joblog in a specific output queue is very inexpensive, this setting can be used all of the time.

Application development: In-house programmers developing new client/server applications may want to track usage of the system as they begin to test the application. By turning on the following functions they can trace the flow of statements to the System i and get an early indication of the performance characteristics of the application:

- Log connection information. This will allow them to run reports to determine the total resource usage for each connection.
- Record interface requests in joblog. Information that is collected by insure/MONITOR will effectively provide trace information of operations that are executed as part of the application. With this information, programmers can ensure the application logic is correct and identify a high amount of network traffic that can lead to performance problems.
- Joblog information. With the use of this feature the programmer can more quickly capture diagnostic information as the application is debugged without having to write specialized debug code or waste time looking for a joblog.

Recommended settings:

- If the product is being used on a test machine, these settings can be used all of the time. If used against a production machine it is recommended that the settings only be used when necessary due to their potential performance impact.
- If the settings are defined for a small number of users then they can be used more often.

Usage monitoring: A System i security officer is probably most interested in an audit trail of user activity. The following features can be used for this purpose:

- Log connection information. The built-in reports provide an easy method to determine who connects to the system and when.
- The interfaces tab allows only “unofficial” software to be audited. This ensures that any attempt to get into the system with unsupported software can be identified (although a better solution to this problem is to define security policies that will prevent the use of specific interfaces altogether).

Recommended settings:

- Logging connection information will have a small impact to sign on and signoff processing. If users stay connected for long periods of time then there

will be little impact. If applications stop and restart connections on a regular basis, it is recommended that the settings be turned on for a small number of users to see what the real impact is before turning them on for all profiles.

- If the settings are defined for a small number of users then they can be used all of the time.

Capacity planning and charge-back: The following setting will help you do capacity planning.

- **Job Accounting.** This feature allows you to define special accounting codes for network applications and users. With existing job accounting software available from a variety of vendors, you can determine the cost of specific applications and groups of users from a client/server perspective. For example, you can define a job accounting code that will automatically get used for green-screen applications and a different accounting code used with an interactive query tool. Because the resources are broken down in a more granular manner, it is easier to determine how costs should be distributed or how much resource a particular application is using.
- **Log connection information.** This setting allows the usage information for a network application to be tracked in a similar manner to Job Accounting, but with a “network access” flavor. You can track the increased usage for a particular user or application over time and project the future impact if that application is deployed to more users.

Recommended settings:

- If you have a job accounting software package, or are currently using job accounting with homegrown reports, you should use the job accounting feature to obtain more granular data.
- If you want to isolate and report on the resources required for network applications, use the Log connection information. As mentioned before, if applications connect and disconnect frequently, there will be some performance penalty associated with connection logging, but in most cases the impact should be minimal.

Resource and security policy definition: You may want to use insure/MONITOR to collect information about current activity before defining policies. This allows you to better understand your user environment in order to control it most effectively. The following features of insure/MONITOR can help you:

- **Record interface requests in joblog:** Turning the logging feature on for your network users will result in activity being collected for the real user profile.
- **Logging connection information:** By collecting connection information you can determine the “normal” working hours of system users and put proper time-frames on security, resource, and tracking policies.

Recommended settings:

- To track resource usage at a user level, just enable connection logging. The reports that come with insure/MONITOR will provide information about user activity based on this level of tracking.

4 Scenarios for product use

4.1 Using insure/MONITOR to support users

This section describes several common support issues and a step-by-step approach to address them using insure/MONITOR.

4.1.1 Introduction

4.1.1.1 Issues

Applications based on network protocols such as TCP/IP, ODBC, DRDA, and DDM introduce unique problems to IT organizations that must support those applications and the users who depend on them.

The first issue is caused by the basic architecture of a network application. The implementation of client/server or Internet protocols requires the use of System i jobs that are not easily associated with a specific end-user. These jobs perform work on behalf of attached clients and are not named the same way traditional green-screen jobs are. Typically, these jobs are run under a generic user profile such as QUSER when they are created and are later associated with a user when needed. As a result, support personnel find it difficult to work with user profile information using traditional diagnostic techniques.

The second issue is the complexity of the application environment. Typically, network applications are built on top of complex protocols. The application programmer is often insulated from many details of data conversion, communication flow, and database access. While this insulation results in much better programmer productivity, it often means that the programmer may not understand what happens “under the covers”.

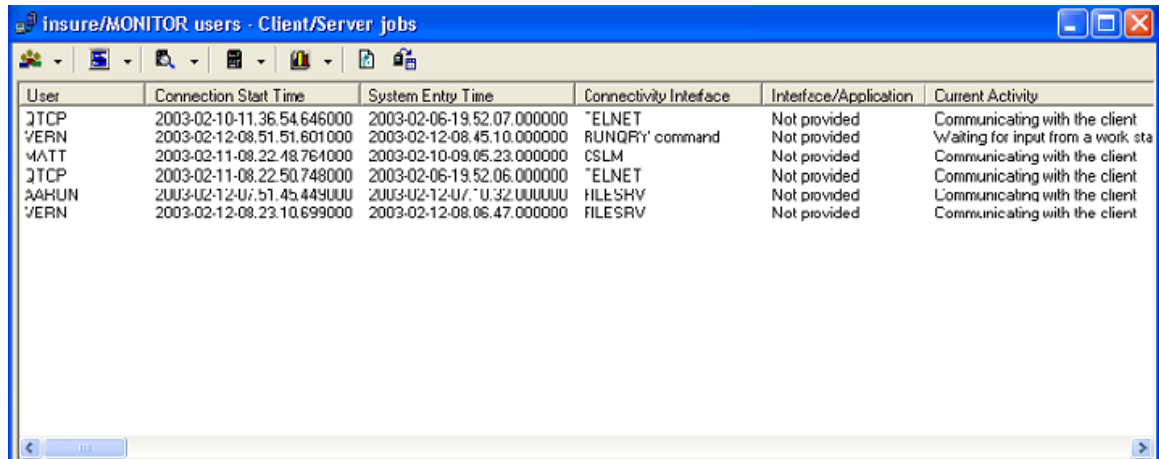
4.1.1.2 Challenges

The challenge with these environments is to manage users and applications effectively. insure/MONITOR lets you build on existing knowledge and diagnostic approaches while supporting these non-traditional environments. Specifically, insure/MONITOR gives you:

- The ability to identify network users by real user profile rather than generic user profile.
- Familiar diagnostic information (available through WRKACTJOB) with new information such as the last operation attempted and network statistics.
- Point-and-click access to jobs, related information, and tools to diagnose and fix problems.

4.1.2 Common Tasks with the User Monitor

The following sections discuss several different tasks that can be performed using the User Monitor window in insure/MONITOR. The main window for the User Monitor looks similar to the following:



User	Connection Start Time	System Entry Time	Connectivity Interface	Interface/Application	Current Activity
JTCP	2003-02-10-11.36.54.646000	2003-02-06-19.52.07.000000	*ELNET	Not provided	Communicating with the client
VERN	2003-02-12-08.51.51.601000	2003-02-12-08.45.10.000000	RUNQRY* command	Not provided	Waiting for input from a work sta
MATT	2003-02-11-08.22.48.764000	2003-02-10-09.05.23.000000	CSLM	Not provided	Communicating with the client
JTCP	2003-02-11-08.22.50.748000	2003-02-06-19.52.06.000000	*ELNET	Not provided	Communicating with the client
AAHUN	2003-02-12-08.23.10.699000	2003-02-12-08.06.47.000000	FILESRV	Not provided	Communicating with the client

The User Monitor allows you to view all client-server activity currently on your system. You can select horizontal jobs filter by clicking on the arrow next to the first button. The possible options are all jobs, jobs just using system resources, client/server jobs, OneWorld jobs (NOTE: functionally limited to capturing of the group of OneWorld jobs thus it is not recommended at this time).

The second button provides a vertical filter to select which columns you want displayed in this view.

The third and fourth buttons provide the ability to directly affect selected jobs (i.e. hold, release, end, retrieve attributes etc.).

The fifth button provides a visual representation of the job statistics in form of bar/pie charts.

The sixth button is for manual refresh and seventh provides ability to setup automatic refresh.

Note: If you also have a license to Centerfield Technology's insure/RESOURCES tool, you will see entries for the interfaces covered by that tool: OPNQRYF, RUNQRY, ODBC/JDBC, STRSQL, and RUNSQLSTM. If you do not have a license to insure/RESOURCES, you can get this same information in the User Monitor by running the ADDQCEXIT command from the {PROGRAM LIBRARY} library on your System i server.

Associating System i Job name to actual user: To determine which System i job is being used by a network application user, simply scroll to the right until you find a column named “Qualified Job Name”. This is the same job name you will see by doing a WRKACTJOB on the System i. The “User” column indicates which user profile is currently associated with that job.

Looking at a System i joblog: To look at the joblog for any of the connected users, just double-click the desired job. Once you have the joblog, you can easily search the data returned to find information of interest (something you can’t do on the equivalent System i display). You can also print the joblog or cut and paste a portion of it to another application.

Determining the last operation: The last operation field holds information not available from a green screen. In the case of ODBC applications it contains the last SQL statement executed by the desktop application. This allows the IT support person to determine what activity happened last so they can answer questions or diagnose problems.

Determining the application and activity: Along with traditional information such as CPU time, insure/MONITOR allows you to determine what type of interface the network application is using. It also allows you to determine what the current state of the job is to aid in the diagnosis of problems and their source. For example, if you have a user with a PC application which is “hung”, you can determine if the problem is on the System i or the PC by looking at the activity field, system resource indicators, and last operation fields.

Change job and query attributes: In problem diagnosis or resolution situations, it is desirable to be able to change job attributes to assist in these activities. insure/MONITOR allows you to easily change basic job and query attributes. In a situation where an application is taking a high percentage of CPU time, you might want to reduce the job’s priority and timeslice.

4.1.3 Tips and techniques

This section contains tips and techniques you can use with various types of problems with end-user support.

Runaway query:

If you have a query that is taking a large percentage of the CPU, you can do one or all of the following:

- Reduce the job’s run priority
- End the job or put the job on hold
- Change the job’s query attributes to prevent another runaway query. (This will not do

-
- anything to stop the current query from running.)
 - Diagnose the problem

If you choose to diagnose the problem here is a step-by-step approach to gather information about the job:

1. Highlight the job causing the problems
2. Look at the last operation attempted. If the last operation was an SQL statement
3. Double click on the call stack. The following list indicates what the likely operation is if you see the named modules at the bottom of the call stack and the job is running with very high CPU:

QDBGETM	Records are being fetched. It is possible that a table scan is occurring.
QQTSORT	The query optimizer has chosen to sort records instead of using an index.
QQQIMPLE	An index build is occurring.

4. Click on the Change Query Attributes action. On the next dialog, turn on Query logging by checking the box that says to include SQL and query messages in the joblog. Subsequent queries submitted by that user will now record additional information in the joblog so you can look at those messages.

Slow query response time:

To diagnose slow query response times, perform the following steps:

1. Find the user who is having the problem.
2. Click on the Change Query Attributes action and turn on SQL and query messages in the joblog.
3. Have the user re-run the query that is performing poorly.
4. Double-click on the job or choose Joblog from the list of Actions. Once you see the joblog you will be able to see what message the query optimizer put in the joblog for this particular query.
5. You can now take one of the following actions:
 - Make recommendations to the end user based on your past experience on how to best formulate a query given your data and system configuration.
 - Use Visual SQL Explain to understand the query in greater depth and experiment with different options before making recommendations to the end-user. The help text for Visual SQL Explain gives extensive information about query implementation and ways to influence the System i optimizer to achieve better performance.

- Use insure/INDEX to determine if building an index would help the performance of the problem query.

Unknown error:

If a user reports a failure in a query or network application, and the messages they receive do not help you determine the true cause of the problem, take the following steps:

1. Examine the user's joblog by double clicking on the user's job.
2. If there is still not enough information in the joblog, and you suspect that the problem is related to something happening on the System i, enable query and SQL messages by using the Change Query Logging function.
3. Look at the job's library list if it appears the problem is related to not finding a particular file or other object.
4. The other information which might shed light on a problem are:
 - Locks
 - Open Files
 - File override information

If there is still a problem getting enough information, do the following:

1. Start the policy editor. This is done by choosing the menu options: Tools->insure/MONITOR->Monitor policies.
2. Identify the user or group you want to diagnose problems for and choose the following options:

Setting	Purpose
Log Connection Information	Used by the tracking reports to identify when users enter and exit the system.
Record interface requests in joblog	Ensures that the joblog will contain the maximum amount of information necessary to debug functional and performance problems.
Create joblog	Ensures that a joblog is produced and is put into a known output queue.

3. Once the settings have been defined, have the user restart their application and re-connect to the System i. Their new connection will collect the information you specified in step 2.



5 Support

Centerfield Technology Inc. is committed to providing our customers with support as problems or questions arise. Support includes minor enhancement releases and upgrades which might be necessary as OS/400 and i5/OS releases become available.

5.1 Contact Information

Internet

Web support pages are maintained at www.centerfieldtechnology.com. Updated documentation, how-to's, FAQs, and a list of known problems will become available as needed.

E-mail

To contact technical support by email, send email requests to support@centerfieldtechnology.com.

Fax

To contact technical support by fax, send your request to **(888) 908-3073**.

Phone

You can call 507-287-8119 for technical support. Telephone support is available from 8:00 AM to 5:00 PM CST.

5.2 Additional Information

- Visit Centerfield Technology's HomeRun web site www.centerfieldtechnology.com to find the latest software patches for HomeRun. Follow the support link from the home page.
PTFs listed in this document are current as of the date this document was released. There are times where IBM releases PTFs applicable to the functions our software uses after the document release date. Accordingly, we strongly recommend you visit our website periodically and verify that all listed PTFs are applied on your system.

6 Appendixes

6.1 Lock detection and diagnosis

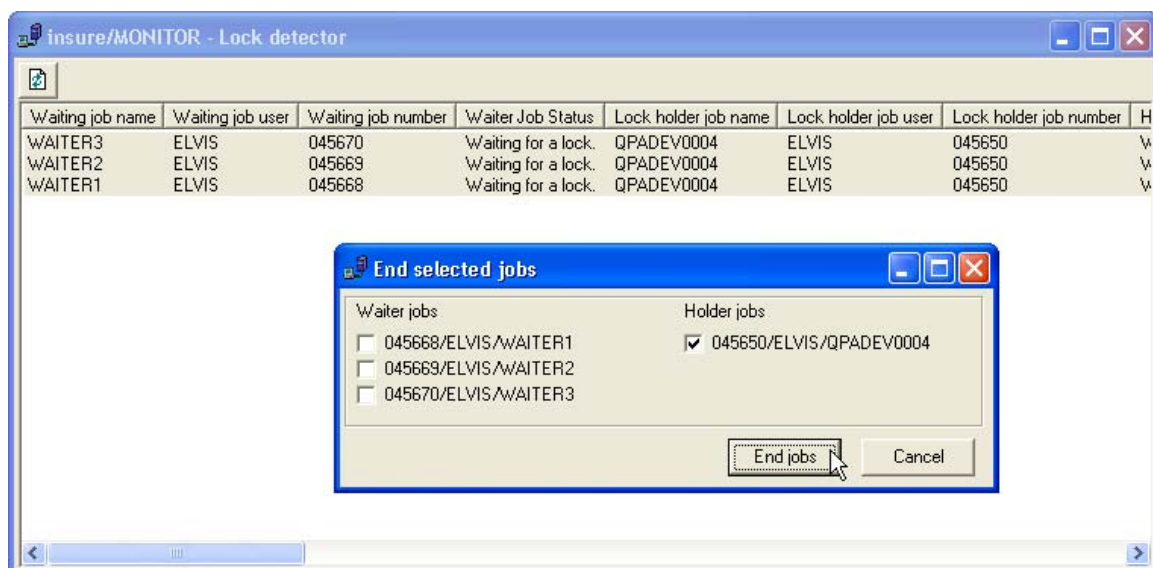
When a lock problem surfaces on a System i, it is not straightforward to find the root cause of the problem. Administrators must sort through hundreds (or thousands) of jobs in an attempt to identify the type and ownership of the lock. Once that task is complete, action still can't be taken until the relationship between type, ownership, and process is understood.

Insure/MONITOR eliminates flipping through multiple green screens to guess at who and which processes are causing the problem. With a simple GUI, insure/MONITOR pinpoints locking issues and provides the information needed to take quick action so other contingent business processes are not affected.

With a single click you can determine:

- What kind of lock is being waited for (object or record lock)
- Which job/user owns the lock being requested
- What action to take once the root cause has been identified

Here is a screen shot of the lock support in action:



6.2 Semaphore Wait Detector

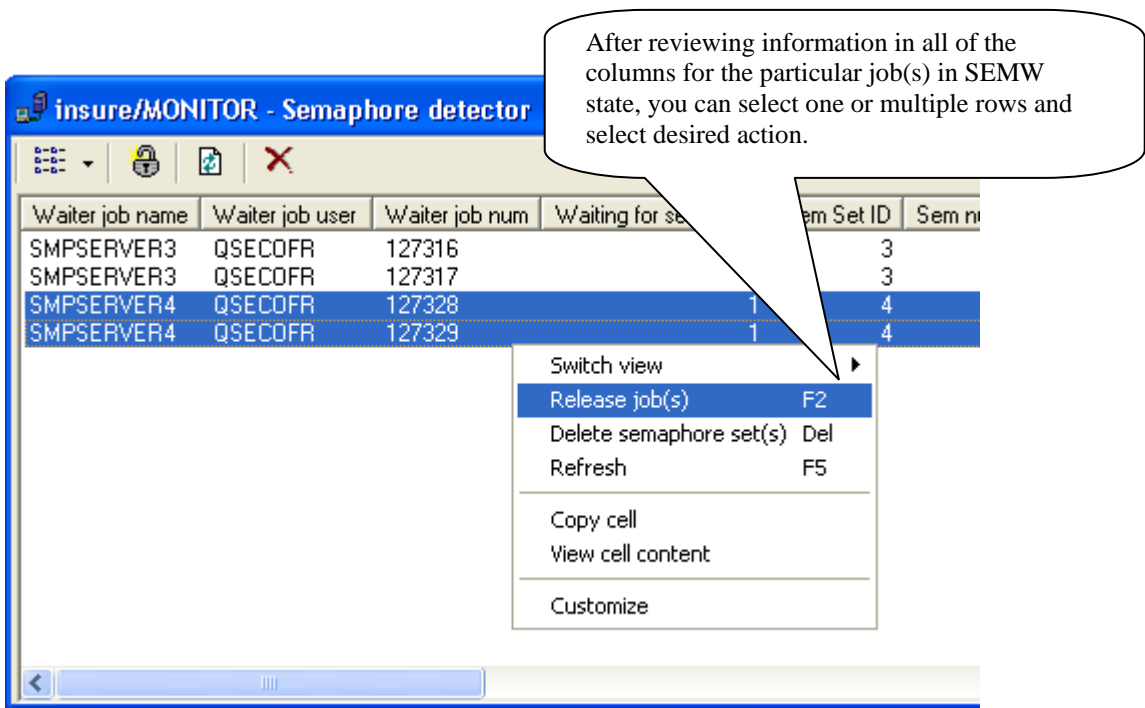
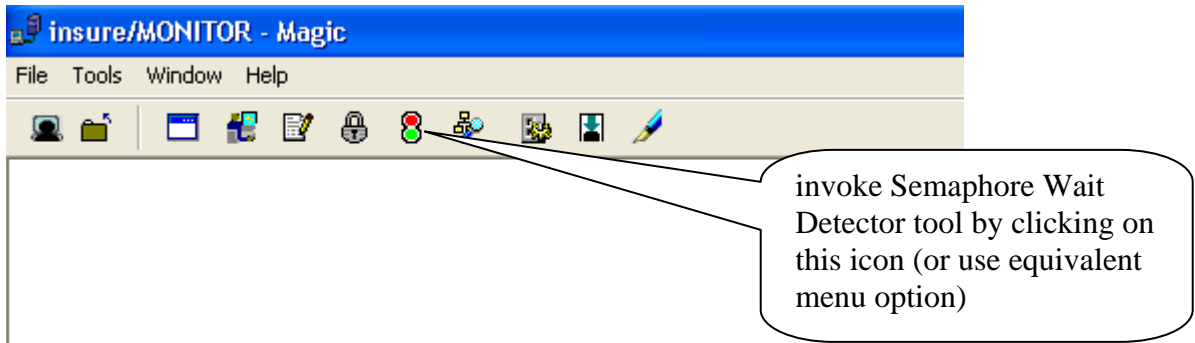
Many applications ported from other platforms (i.e. UNIX) or even native to the System i resort to using semaphores as a locking or resource allocation mechanism. With these applications it is common for an administrator to see System i jobs in a SEMW (semaphore wait) state. This is a normal state of a job waiting for a certain semaphore to reach a logical value before the job proceeds with its next action.

There are, however, times when this state is not normal. An abnormal SEMW state can be entered into due to application oversight, resource conflict or operator error. In these cases, it would be desirable for the system administrator to obtain detailed information about the semaphore being waited on and take action to resolve the conflict. Possible actions are posting desired wait value to the semaphore, in effect releasing the job from the semaphore wait state. Or alternatively, if administrator concludes that the owning semaphore set is orphaned or damaged, he/she can decide to delete the orphaned semaphore set.

With 5.2 version of insure/MONITOR, the Semaphore Wait Detector tool lets you do exactly that by displaying all of the jobs waiting for semaphores and the associated semaphore information and letting you take action to resolve the conflicts.

Since semaphores are inter-process communication objects, it is easy to conceive a situation where one faulty job can cause other jobs to erroneously enter the SEMW state. Rather than holding up multiple jobs, administrators can now make well informed decisions and opt to take action by posting the value being waited on or deleting the damaged semaphore set.

Following screen shots demonstrate how to use the Semaphore Wait Detector tool.



Release jobs in SEMW state by posting their wait values

Semaphore post function allows system administrators to take action on jobs in the SEMW (semaphore wait) state in an effort to release the necessary resource and let the required work continue normal operation. Some causes of non-designed queuing of jobs in SEMW status are resource conflicts, application programming oversight, operator error etc.

To modify the number of released jobs click on the highlighted cell(s). Jobs will be released in FIFO (First In First Out) order.

Semaphore Set ID	Semaphore Number	Wait Value	Release first N	or MAX jobs
4	0	1	1	2

Post Cancel

To modify number of jobs to release, click inside the cell. If multiple jobs are waiting on the same semaphore value, they're released in FIFO order.

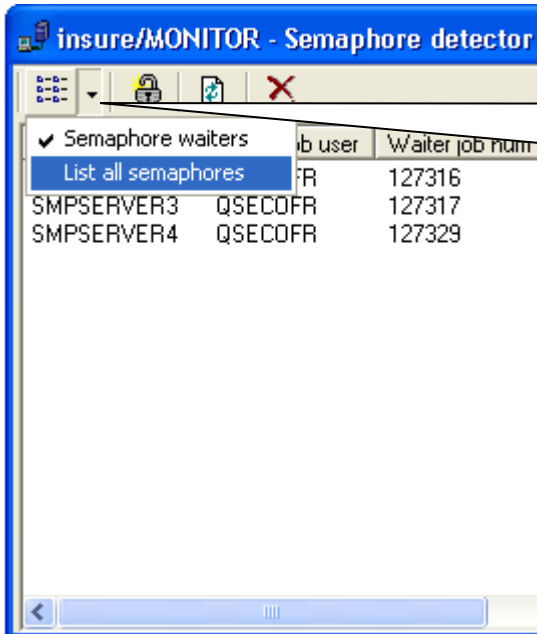
Once the number of jobs to release is set, click Post button to post the value being waited on to the appropriate semaphore number.

insure/MONITOR - Semaphore detector

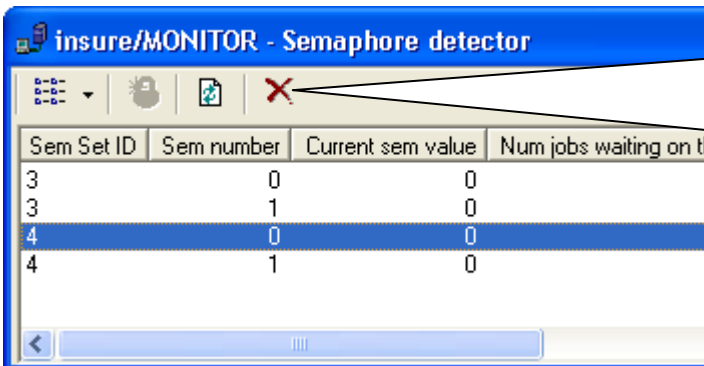
Waiter job name	Waiter job num	Waiter job num	Waiting for sem value	Sem Set ID	Sem
SMPSERVER3	QSECOFR	127317	1	3	
SMPSERVER3	QSECOFR	127317	1	3	
SMPSERVER4	QSECOFR	127329	1	4	

Switch view
Release job(s)
Delete semaphore set(s) Del
Refresh F5
Copy cell
View cell content
Customize

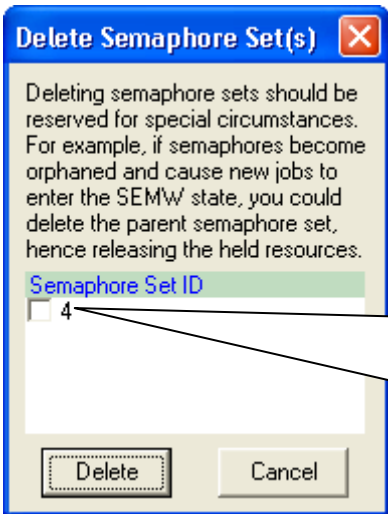
Give the waiting job(s) few seconds to become cognizant of the fact that the appropriate SEMW value has been posted and refresh the screen by one of 3 methods (icon, F5, or right-click menu). As you can see, one of the jobs has been released from the SEMW state.



To view raw list of the existing semaphore sets and associated semaphores, invoke a drop down menu off of this icon and select "List all semaphores" menu option.



If after reviewing the presented information you conclude that the semaphore set is damaged or orphaned, you can delete the semaphore set in question by one of three methods (X icon, Del key, right-click menu option).



Selected semaphore set id(s) will be presented in this dialogue. For added safety, user is required to check the deletion checkbox to ensure that is truly the desired action for the semaphore set(s). Upon that, hit Delete button to execute the action.

6.3 Mutex Wait Detector

Many applications ported from other platforms (i.e. UNIX) or even native to the System i resort to using mutexes for locking and synchronization purposes. With these applications it is common for an administrator to see System i jobs in a MTXW (mutex wait) state. This is a normal state of a job waiting for a certain mutex to be released for use. After mutex is obtained, waiting job proceeds with its next action.

There are, however, times when this state is not normal. An abnormal MTXW state can be entered into due to application oversight, resource conflict, operator error, incorrect installation (deployment) etc. In these cases, it would be desirable for the system administrator to obtain detailed information about the mutex being waited on and take action to resolve the conflict. Possible actions are ending the job that is holding the mutex since that holder jobs may be in error and will never release the mutex through normal application means. Alternatively, armed with the additional information about the cause of the mutex wait, administrator can contact application developer or the ERP vendor support staff.

With 5.3 version of insure/MONITOR, Mutex Wait Detector tool lets you do exactly that by displaying all of the jobs waiting for mutex and the associated mutex information and letting you take action to resolve the conflicts.

Since mutexes are inter-process communication objects, it is easy to conceive a situation where one faulty job can cause other jobs to erroneously enter the MTXW state. Rather than holding up multiple jobs, administrators can now make well informed decisions and opt to take action by ending the erroneous job or arm themselves with more details on the mutex before contacting development support personnel.

Following screen shots demonstrate how to use the Mutex Wait Detector tool.



You will be presented with a window listing all jobs presently in the MTXW state. If the window is blank, there are no jobs in MTXW state presently on your system. If there are jobs showing, analyze for the scenarios where there is a single mutex holder (columns 7-10) with several jobs waiting for the held mutex to be released.

Waiter job n...	Wait...	Wait...	Waiter thread ID	Mutex Name	Mutex State	Owner job n...	Owner job...	Own...	Owner thread ID	Num of wai...	Owner thread value
MTXWAIT1	JDE	094318	0000000000000006	UNNAMED_MCRT	Thread waiting for mutex	MTXHOLD	JDE	094317	0000000000000009	3	B003200004E89000
MTXWAIT2	JDE	094319	0000000000000016	UNNAMED_MCRT	Thread waiting for mutex	MTXHOLD	JDE	094317	0000000000000009	3	B003200004E89000
MTXWAIT3	JDE	094320	0000000000000008	UNNAMED_MCRT	Thread waiting for mutex	MTXHOLD	JDE	094317	0000000000000009	3	B003200004E89000

Besides observing mutex and holder/waiter job information and passing it on to the development support personnel, administrators can take action to resolve the situation. The common action would be to end the holder job.

Mutex Name	Mutex State	Owner job n...	Owner job...	Own...	Owner thread ID	Num of wai...	Ow
UNNAMED_MCRT	Thread waiting for mutex	MTXHOLD	JDE	094317	0000000000000009	3	BO
UNNAMED_M		MTXHOLD	JDE	094317	0000000000000009	3	BO
UNNAMED_M		MTXHOLD	JDE	094317	0000000000000009	3	BO

With current trend of ERP applications migrating from other platforms to the System i, usage of mutexes for inter-process synchronization will increase. Centerfield's mutex/DETECTOR will aid the System i administrator in problem diagnosis.

6.4 Historical data archive (CFGMONHST)

6.4.1 Overview

Large portions of the current customer base of insure/MONITOR has expressed interest in having data normally displayed in our PC client GUI windows stored in a physical file on the System i for historical and archiving purposes.

One real-life scenario is where a customer was having issues with extensive page faults on certain jobs and was able to track down the job names using native system facilities. However, jobs turned out to be ODBC jobs (QZDASOINIT) in a shop running Oracle's EnterpriseOne (Oneworld) ERP application and even though the administrator was able to track the jobs down, he still didn't know who the users were and what SQL statements they were running that caused excessive page faulting.

With insure/MONITOR logging this type of data and availability of a summary view based on the Citrix server, time of day, and aggregated page faults per job, a system administrator is now able to track down who and what is causing excessive page faulting.

6.4.2 Configuration and setup

The archiving feature is **not** turned on by default. To turn it on you will need to use green-screen CFGMONHST command, located in *{PROGRAM LIBRARY}* library. The command comes with online help (hit F1) but I'll explain its parameters briefly here:

- **COLLECT**
 - specify if you want to turn on historical data archiving
 - data is stored in XCMONHST file
 - if *YES is selected, collection job starts automatically
 - there are a number of summary views and reports available for your convenience (see the section following this one). We strongly encourage end-users to use these reports and views as templates that can be customized for your particular needs.
- **JOBFILTER**
 - Collect on all system jobs (i.e. WRKACTJOB view)
 - Collect on client/server jobs only – which jobs are actually collected is dictated by which of Centerfield's exit point programs are registered on your system. Registering Centerfield exit point programs is done via *{PROGRAM LIBRARY}/ADDMON* command and can cover around 20 different exit points (ODBC, FTP, file server, DRDA etc.).
 - Collect on EnterpriseOne (Oneworld) only jobs. This includes JDENET_K kernel jobs, SENTINEL, NETWORK, UBE – universal batch

-
- engine jobs (based on UBEPREFIX setting) and ODBC jobs we have captured the Oneworld (Citrix) user profile for.
- Collect on all jobs that have been active within the past 15 seconds (used CPU resource).
 - Collect on all system Oneworld jobs (as outlined above) as well as all ODBC jobs irregardless if they're coming via Oneworld ERP application, Microsoft Access or some other JDBC/ODBC interface.
 - Collect for jobs that have used system resources in the past 15 seconds. This option provides information about active jobs while consuming a much smaller amount of disk space than the other options.
- **REFRESH**
 - Setting controlling how often data is collected
 - **PURGE**
 - Setting controlling how long data is kept in the historical archive before being purged
 - **UBEPREFIX**
 - We assume by default that EnterpriseOne (Oneworld) UBE (Universal Batch Engine) jobs are System i batch jobs and that the naming convention is that they start with the letter R, i.e. R*). However, we also realize that each shop can and does customize this to match their internal business rules. To help us determine what are UBE jobs, set this setting appropriately, i.e. R* or FW* or whatever else uniquely identifies UBE jobs.
 - **HSTLSTRFSH**
 - Historical archive collection will attempt to resolve IP addresses to canonical (official) computer names as they're defined in your LAN or in the DNS table. If host name resolution is successful, collection will store these computer names into RMTLOC field of the XCMONHST file. In typical environments computer names assigned to certain IP addresses don't change frequently. Due to that fact and to minimize collection overhead of calling host name resolution function, we store resolved names in the keyed user index objects and on subsequent file writes just perform a look up on that index. By default, we refresh the user index list once an hour. You can control this refresh interval setting by via this command parameter (HSTLSTRFSH).
 - NOTE: as a side effect of changing this setting, our archival code will recreate the keyed user index and start the interval count from that moment.

Besides the main, top-level job filter defined with the JOBFILTER parameter on CFGMONHST command, we also allow to further filter out jobs that are being collected on by using ADDJOBFCFG, RMVJOBFCFG and CHGJOBFCFG commands. These three commands act on the subset of jobs already filtered by the JOBFILTER parameter. We recommend using **CHGJOBFCFG** command to deal with these filters, as it will show

you any existing job name filters.

Filters are stored in XCJOBFCFG physical file, so you could also see them by running RUNQRY () /DATA LIBRARY/XCJOBFCFG command.

Let me illustrate couple of scenarios where this additional job name filter could prove useful:

- Let's say you want to monitor interactive jobs only. You could define JOBFILTER to *ALL and then use ADDJOBFCFG to add a QPADEV* filter to log historical data about all interactive jobs (assuming default naming convention for interactive devices is used).
- Another example may be where user didn't register any of our exit points via ADDMON command, but would still like to get historical statistics on the JDBC/ODBC jobs. In this scenario, you can define a JOBFILTER with *ALL and then ADDJOBFCFG for QZDASOINIT jobs.

Tip: consider adding a trigger program to XCMONHST file that would examine certain conditions (i.e. page faults delta) and notify you if it exceeds certain predefined value.

6.4.3 Archive reports and views

This document describes the reports in insure/MONITOR that access the information deposited in the *{DATA LIBRARY}/XCMONHST* file when the auditing controls are turned on via CFGMONHST command (COLLECT parameter).

6.4.3.1 Overview

The built-in reports utilize a combination of SQL views and Query Manager reports. The SQL views collect columns about a particular topic, summarize data, and perform calculations necessary for a particular set of reports. The Query Manager reports provide runtime prompting to do data selection, sorting, and “top N” information for summary analysis and problem identification. The design idea behind these SQL views and Query Manager reports is to serve as the template foundation for further end user customization.

6.4.3.2 Naming conventions

The SQL views and Query Manger reports in the Centerfield library follow a naming convention to help identify their purpose. The following is the key to that naming convention:

Position 1: S = Summary, D = Detail

Position 2: F = Page Faults, T = Temporary Storage, C = CPU, I = Disk I/O, N = number of active users, W = waits, A = All metrics

Position 3: U = runtime selection on User, D = runtime selection on Date, Time, and User R = recent metrics (e.g. today), N = no selection

Position 3-10: String to further describe the report or view

For example, a Query Manger report named SARBYHOUR provides a summary of page faults by hour just for the current day.

You can also look at the object description of each object to see information about its use.

6.4.3.3 Reports

Name	Purpose
DANMETRICS	Base report that contains all performance metrics and identification information. Intended to be used as a template for other, more specific reports.
DARMETRICS	Base report that contains all performance metrics and identification information <i>for the current day</i> . Intended to be used as a template for other, more specific reports.
DCDCPU	CPU metrics with job and user identification information. Runtime prompting for a specific user and date/time range. Only top 100 jobs returned.
DCNCPU	CPU metrics with job and user identification information. Top 100 jobs returned over all currently collected data.
DCRCPU	CPU metrics with job and user identification information. Top 100 jobs returned over all current days activity.
DFDPGFLT	Page fault metrics with job and user identification information. Runtime prompting for a specific user and date/time range. Only top 100 jobs returned.
DFNPGFLT	Page fault metrics with job and user identification information. Top 100 jobs returned over all currently collected data.
DFRPGFLT	Page fault metrics with job and user identification information. Top 100 jobs returned over all current days activity.
DIDIO	Disk I/O metrics with job and user identification information. Runtime prompting for a specific user and date/time range. Only top 100 jobs returned.
DINIO	Disk I/O metrics with job and user identification information. Top 100 jobs returned over all currently collected data.
DIRIO	Disk I/O metrics with job and user identification information. Top 100 jobs returned over all current days activity.
DTDTMPSTG	Temporary storage metrics with job and user identification information. Runtime prompting for a specific user and date/time range. Only top 100 jobs returned.
DTNTMPSTG	Temporary storage metrics with job and user identification information. Top 100 jobs returned over all currently collected data.
DTRTMPSTG	Temporary storage metrics with job and user identification information. Top 100 jobs returned over all current days activity.
DWDWAITS	Wait time metrics with job and user identification information. Runtime prompting for a specific user and date/time range. Only top 100 jobs returned.

DWNWAITS	Wait time metrics with job and user identification information. Top 100 jobs returned over all currently collected data.
DWRWAITS	Wait time metrics with job and user identification information. Top 100 jobs returned over all current days activity.
SANBYHOUR	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on an hourly basis.
SANBYHOURI	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on an hourly basis <i>by IP address</i> .
SANBYHOURU	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on an hourly basis <i>by OneWorld user</i> .
SANBYMIN	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on a minute basis.
SANBYMINI	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on a minute basis <i>by IP address</i> .
SANBYMINU	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on a minute basis <i>by OneWorld user</i> .
SARBYHOUR	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on an hourly basis for the current day.
SARBYHOURI	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on an hourly basis <i>by IP address</i> for the current day.
SARBYHOURU	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on an hourly basis <i>by OneWorld user</i> for the current day.
SARBYMIN	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on a minute basis for the current day.
SARBYMINI	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on a minute basis <i>by IP address</i> for the current day.
SARBYMINU	Summary report for number of concurrent users, page faults, cpu, disk i/o, and temporary storage summed on a minute basis <i>by OneWorld user</i> for the current day.

When you initially start using QM query objects to run these sample reports, the best option would be to use STRQM command like:

- 1) ADDLIBLE {*DATA LIBRARY*}
- 2) STRQM
- 3) 1
- 4) type {*PROGRAM LIBRARY*} for the library and hit Enter
- 5) select option 9 to Run the QM query
 - o make sure arguments are enclosed with apostrophes for queries that require input parameters

As you learn more about available QM queries and familiarize yourself with invoking them, you might want to call them directly, without invoking STRQM first, i.e.:

```
STRQMQRV QMQRV({PROGRAM LIBRARY}/SARBYHOUR)
```

When directly calling QM queries that require user input, make sure to enclose input arguments with triple quotes, i.e.:

```
STRQMQRV QMQRV({PROGRAM LIBRARY}/DCDCPU) SETVAR((USERPROFILE  
"Mary") (STARTDATE "'10/01/05'") (STARTTIME "'00:00:01'") (ENDDATE  
"10/13/05") (ENDTIME "'23:59:59'"))
```

You can embed these queries into CL programs and pass CL variables to the queries, i.e.:

```
PGM (&usrprofile &startdate &starttime &enddate &endtime)  
DCL &USRPROFILE *CHAR 10  
DCL &STARTDATE *CHAR 8  
DCL &STARTTIME *CHAR 8  
DCL &ENDDATE *CHAR 8  
DCL &ENDTIME *CHAR 8
```

```
STRQMQRV QMQRV({PROGRAM LIBRARY}/DCDCPU) +  
SETVAR((USERPROFILE ("' *TCAT &USRPROFILE *TCAT '")) +  
(STARTDATE ("' *TCAT &STARTDATE *TCAT '")) +  
(STARTTIME ("' *TCAT &STARTTIME *TCAT '")) +  
(ENDDATE ("' *TCAT &ENDDATE *TCAT '")) +  
(ENDTIME ("' *TCAT &ENDTIME *TCAT '"))))
```

```
ENDPGM
```

This could prove useful if you select *OUTFILE option for OUTPUT parameter, as the output file can then be read by RCVF CL command.

All of the QM query objects run SQL statements against the underlying SQL views built on top of base XCMONHST physical file.

To see all available views, do:

```
DSPDBR {DATA LIBRARY}/XCMONHST
```

You can query these views directly via SQL or other methods (i.e. RUNQRY, OPNQRYF, RPG etc.).

As these SQL views and QM queries are intended to serve as templates, end user should create their own views and QM queries. Most of the times only minor tweaks to template SQL statements will be required. In this scenario end user may want to view base SQL and use it in creating their own views and QM queries. Here are some suggestions on how to do that.

To see existing SQL statements in shipped SQL views do:

```
DSPFD {DATA LIBRARY}/<view name>
```

To see existing SQL statements in shipped QM queries do:

```
STRQM  
1  
{PROGRAM LIBRARY}  
5
```

or to retrieve SQL to a source member:

```
RTVQMQR Y QMQR Y({PROGRAM LIBRARY}/<qm query name>)  
SRCFILE(<your lib>/<your source physical file>)
```

Create your own views using interactive SQL (STRSQL command).

Create your own QM query objects by creating an SQL source member and executing CRTQMQR Y command against it.

6.5 Check Commit Control (CHKCMTCTL)

6.5.1 Overview

Many System i operations that run **24x7** with zero downtime use the operating system Save While Active function in order to provide critical back-ups. This is a good solution but can run into problems when transactions remain open for long periods of time.

A worst-case scenario forces the System i administrator to repeatedly restart Save While Active when stopped due to a stalled commit cycle. Meanwhile, no back up is possible until the long running transaction is identified and addressed.

In order to help customers pro-actively deal with this problem, Centerfield Technology has created a Check Commit Control feature that detects when a database transaction (commit cycle) exceeds a user-defined amount of time. Whenever a long-running transaction is detected, a message is sent to a queue, and an event is logged in a database file.

Identification of long commit cycles can help isolate application problems and provide early detection and notification when open transactions might prevent critical operations, like backups, from running successfully.

6.5.2 Configuration and setup

A straightforward way to utilize this feature is to simply submit a job that invokes the command with the settings that suite your environment. The command comes with online help (hit F1) but I'll explain its parameters briefly here:

- **NUMCHKS**
Specifies the number of times the command will loop looking for a long-running transaction. The time to wait between each iteration is controlled by the sample time (SMPTIME) parameter.
Default of *ONCE only checks for a long transaction one time. This setting is intended for interactive command use.
Specify special value *FOREVER to loop until the job running the command is ended. This option is intended to be used in a batch job to continuously monitor a system for transactions that may not complete and send messages to the predefined message queue when user-defined thresholds are reached.
Alternatively, specify number of times to perform the check for a long running transaction before command exits.
- **SMPTIME**
Specifies the delay time interval between checking for long running transactions.

-
- Default is to check every 5 minutes (300 seconds).
- **MAXSEC**
This parameter is used to identify transactions that are considered to be long-running. If a commit cycle exceeds this number of seconds it will be identified and the actions defined by the OUTPUT parameter will be taken (notification, logging or both).
Default of one hour (3600 seconds) is assumed to be a long running transaction. You should modify this default value to better suit your specific business rules.
 - **EXLSYS**
This parameter allows you to exclude system jobs from analysis. If only application databases are of interest, then system jobs should be excluded. Our testing has shown that some system jobs maintain very long commit cycles so we recommend you leave this set to default of *YES.
 - **JRN**
The JRN parameter specifies the name of the journal that will be used to filter the transactions considered by this command. We recommend specifying a specific journal rather than using *ANY to achieve the best performance and ensure that actions are taken within the time period needed.
 - **OUTPUT**
The output parameter determines where notification is sent when a long running transaction is found.
 - By default a message is sent to a system operator message queue QSYSOPR. Most shops have either developed in-house monitoring programs for QSYSOPR message queue or have purchased 3rd party tools to monitor system operator messages. These tools can key off our message and page or email system operator when the long running transactions are encountered.
 - Alternative output is to a file. Information would be logged to XCCHKCMT file in {DATA LIBRARY} file.
 - If you see a need for both types of output, select special value *BOTH.

6.5.3 Examples for CHKCMTCTL

Example 1: Simple Command Example

CHKCMTCTL

This command will immediately look for commit transactions that were started more than one hour ago that still have uncommitted changes. It will ignore system generated transactions, and send a message to the QSYSOPR message queue if those transactions are found.

Here is sample of a formatted message put on the message queue of your choice by the CHKCMTCTL command. Message ID XCC1000 would be the key used in the QSYSOPR monitoring tool to send a page or an email as a warning.

*Message ID : XCC1000 Severity : 40
Message type : Information
Date sent : 09/09/05 Time sent : 16:22:16*

*Message : Job 095973/MLH/QPADEV0010 has exceeded the allowable open
commit transaction time.
Cause : Job 095973/MLH/QPADEV0010 has an open commit cycle that has
started 1276 seconds ago. The transaction was started on 05/09/09 (*YMD) at
16:01:00 and uses journal JRN in library MLH.
Recovery : Use the WRKCMTDFN command to commit or rollback the
transaction or end the job to rollback changes.*

Example 2: More Complex Command Example

CHKCMTCTL NUMCHKS(*FOREVER) SMPTIME(600) JRN(PRODLIB/PRDJRN)
OUTPUT(*FILE)

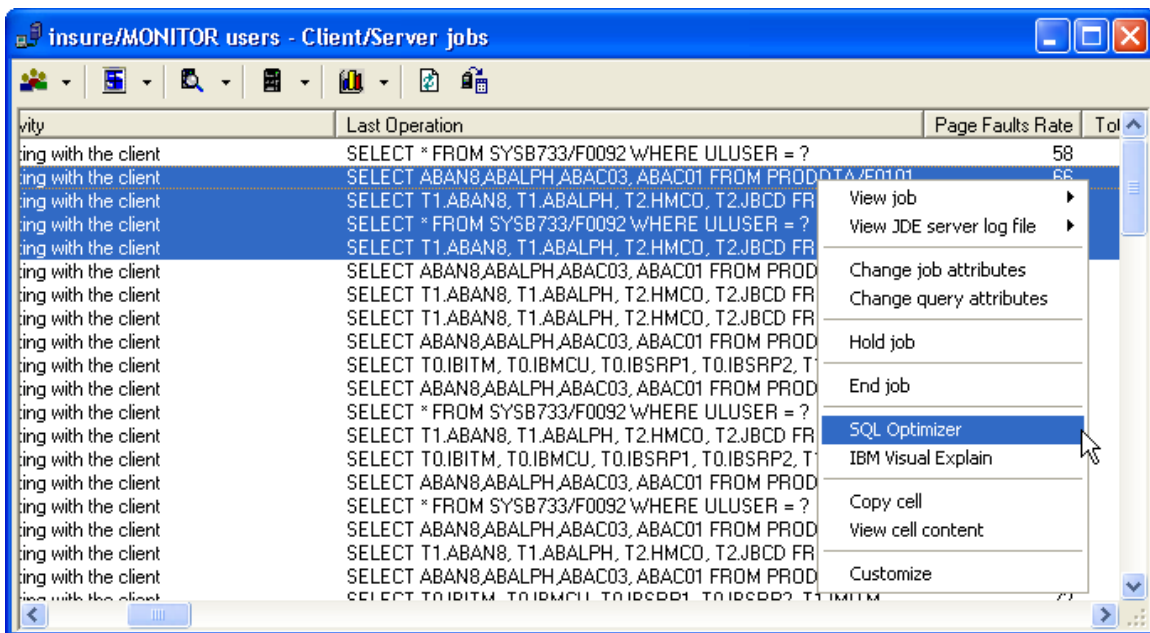
This command will look for long transactions until the job the command runs in is ended. The time between checks is 10 minutes and the command will only look for transactions that are using the PRODLIB/PRDJRN journal. If a long running transaction is found it will be logged to the XCHCKCMT file

6.6 Visual Explain

We have integrated our **Visual Explain** tool with insure/MONITOR. A user can invoke **Visual Explain** from within insure/MONITOR by right-clicking on the cell(s) in the “Last Operation” column and passing the SQL statements located in the cell(s) into the **Visual Explain**.

NOTE: make sure mouse cursor is positioned somewhere over the Last Operation column when you invoke the right-click menu

Here is an example screen capture how to invoke **Visual Explain** passing in the SQL statement:



A detailed look at for Visual Explain (a.k.a. Visual SQL Explain) and usage tips are provided in the next section.

6.7 Using Visual SQL Explain

Visual SQL Explain allows you to quickly understand the implementation of an SQL statement. It makes statement tuning easier by visually showing the actions that the System i query optimizer will use to implement a particular statement and by putting all of the influencing controls at your fingertips. It provides access to the standard SQL debug messages provided by the optimizer and it exposes important statistics related to the implementation path.

6.7.1 Visual SQL Explain plan constructs

The following plan constructs are used when implementing an SQL statement.

6.7.1.1 Sequential (arrival) file access methods



Table scan – Reads all records selecting the ones that match the selection criteria.



Parallel pre-fetch – Reads all of the records in parallel using 2 or more tasks.



Skip sequential processing – Uses a bitmap to selectively read the matching record, reading records in the physical order.



Parallel skip sequential processing – Uses a bitmap and 2 or more tasks to selectively read the matching records.



Encoded vector index load – Processes an EVI index while creating a bitmap.



Parallel encoded vector index load – Processes an EVI index using 2 or more tasks while creating a bitmap.

6.7.1.2 Key field (index based) access methods



Key row positioning – Use an index to selectively read records from a physical file returning the records in order.



Key row selection – Use an index to read all records from a physical file in order.



Index only positioning – Use an index to return the fields selected by the query.



Index only selection – Read the entire index to return the fields selected by the query.

6.7.1.3 Record processing methods



Group by – Process the GROUP BY clause record grouping.



Hash group by – Process the GROUP BY clause using a hashing algorithm.



Sort processing – Process the ORDER BY clause using a memory based sort algorithm.



Distinct processing – Process the DISTINCT clause, returning only unique records.



Record selection – Apply the WHERE clause selection after reading a record.



Union – Combine the results of two or more SELECT statements.

6.7.1.4 Join algorithms



Nested loop join – Join records by reading a record from the first file then finding a matching record in a second file until all matching records are processed in the first file.



Hash join – Join records by first reading all records and organizing them into subsets. Then join matching rows within each subset.

6.7.1.5 Intermediate result processing



Temporary index build – Build an index because an index is required and a suitable index does not exist.



Temporary results file – Build an intermediate file that contains the results up to this point. The rest of the query can operate against these results.



Temporary file – Build a temporary file to hold the results of a view or other complex source file.



Bitmap creation – Build a bitmap to selectively process records.

6.7.1.6 Visual SQL Explain plan constructs reference



Encoded vector index load

Processes an EVI index while creating a bitmap.

This is a good alternative if you do not have SMP capabilities on your system. To influence encoded vector index loading:

Specify the level of parallel processing that is allowed using either the QQRVDEGREE system value, HomeRun, or the CHGQRYA command.



Index only positioning

Index only positioning returns a row or set of rows ordered by a key field. Index only positioning only reads part of an index in a manner similar to a program that uses keys to find a particular record.

Unlike *key row positioning*, it does not do random I/O to access the record in the physical file.

This makes index only positioning the fastest method for accessing an index. This method has the benefits of table scan processing and the benefits of index processing.

See *key row positioning* for implementation tips.



Index only selection

The entire index is read and any selection criteria that references the key columns of the index are applied using the information in the index.

Unlike *key row selection*, it does not do random I/O to access the record in the physical file.

This makes index only selection the fastest method for accessing an index when the entire index must be processed. This method has the benefits of table scan processing and the benefits of index processing.

See *key row selection* for implementation tips.



Key row positioning

Key row positioning returns a row or set of rows ordered by a key field reading. Key row positioning only reads part of an index in a manner similar to program that use keys to find a particular records.

Key row positioning requires a traditional binary radix index.

It is a good alternative for the following cases.

- When a small number of rows will be returned from a file
- When the physical ordering of the records in the file are ordered using the same or a similar order as the key fields in the index.
- When the positioning can be used for join processing and either ORDER BY or GROUP BY processing.
- An existing index can be used instead of a dynamic index build over a large file.

Optimizing index access

- Index access is most efficient when it can do *index only* processing. Criteria for index only processing:
 - All selected fields from the file are contained in the index
 - None of the fields are null-capable
- Specify OPTIMIZE FOR n Rows. If a small number is specified for 'n', the query optimizer will tend to choose a keyed access method more often. If a large number is chosen for 'n', the optimizer will attempt to use a arrival sequence access method if possible. In general it is best to choose a value for 'n' that matches the number of rows you actually expect to be returned from the SQL statement.
- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will allow bitmap based-processing and give the iSeries better statistics about data in the file.
- Remove or modify an ORDER BY clause.
- Change the availability of hash join processing.
- Use the command RGZPFM to organize the physical file to match this index.



Key row selection

This access method requires an index. The entire index is read and any selection criteria that references the key columns of the index is applied against the index.

The advantage of this method is that the table is only accessed to retrieve rows that satisfy the selection criteria applied against the index. Any additional selection not performed through the key selection method is performed at the table level.

The key selection access method can be very expensive if the search condition applies to a large number of rows because:

- The whole index is processed.
- For every key selected from the index, a random I/O to the table occurs.

This is better than key row positioning if every record in the index matches the selection criteria.

Normally, the optimizer will choose to use table scan processing when a search condition applies to a large number of rows. The optimizer only chooses the key selection method if less than 20% of the keys are selected or if an operation forces the use of an index. Options that might force the use of an index include:

- Ordering
- Grouping
- Joining

In these cases, the optimizer may choose to create a temporary index rather than use an existing index. When the optimizer creates a temporary index it uses a 32K page size. An index created using a CREATE INDEX statement normally uses only a 4K page size. The optimizer also processes as much of the selection as possible while building the temporary index. Nearly all temporary indexes built by the optimizer are select/omit or sparse indexes. Finally, the optimizer can use multiple parallel tasks when creating the index. The page size difference, corresponding performance improvement from swapping fewer pages, and the ability to use parallel tasks to create the index may be enough to overcome the overhead of creating an index. Table scans used for building of temporary keyed access paths.

If key selection access method is used because the query specified ordering (an index was required) the query performance might be improved by using one of the following combinations of pre-compiler parameters to allow the ordering to be done with the query sort.

ALWCPYDTA(*OPTIMIZE), ALWBLK(*ALLREAD), and
COMMIT(*CHG or *CS)

ALWCPYDTA(*OPTIMIZE) and COMMIT(*NONE)

Optimizing index access

- Index access is most efficient when it can do *index only* processing. Criteria for index only processing
 - All selected fields from the file are contained in the index
 - None of the fields are null-capable
- Specify OPTIMIZE FOR n Rows. If a small number is specified for 'n', the query optimizer will tend to choose a keyed access method more often. If a large number is chosen for 'n', the optimizer will attempt to use a arrival sequence access method if possible. In general it is best to choose a value for 'n' that matches the number of rows you actually expect to be returned from the SQL statement.
- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will allow bitmap based-processing and give the iSeries better statistics about data in the file.
- Remove or modify an ORDER BY clause.
- Change the availability of hash join processing.
- Use the command RGZPFM to organize the physical file to match this index.



Parallel pre-fetch

Read, using multiple I/O tasks, all of the records from the source table optimizing disk and memory utilization.

If the percentage of rows selected is greater than 20%, this method is more efficient than index processing. On the other hand, if a small percentage of data is selected and the file is large, this access method is not optimal.

Parallel pre-fetch is a good alternative for the following cases.

- A majority of the records in the file will be selected
- There is sufficient main memory with limited contention available for a larger portion of the file in memory at any given time
- The file is distributed across multiple physical disk units. If the file is distributed across multiple disk units the system can perform many physical disk I/O's at the same time which will reduce overall elapse time of the query.

See **Table scan** for information on how to influence parallel pre-fetch implementation.



Parallel skip sequential processing

Uses a bitmap and 2 or more tasks to process records in the order they physically reside in the file, eliminating records that will not be used in the result set as they are read. This approach is preferred to a table scan as it is more efficient.

Parallel skip sequential processing is a good alternative for the following cases.

- A majority of the records in the file will be selected
- The file contains a small number of records
- You have enough main memory available to allow all of the tasks to run efficiently

Influencing Parallel skip sequential processing

- Create an encoded vector index (EVI) for each of the fields use in the WHERE clause for this file. This will improve skip sequential processing efficiency and give the iSeries better statistics about data in the file.
- Specify the level of parallel processing allowed using either the QQRVDEGREE system value, HomeRun, or the CHGQRYA command.
- See *Table scan* for information more techniques.



Parallel encoded vector index load

Processes an EVI index while creating a bitmap using 2 or more tasks.

This is a good alternative if you have SMP capabilities on your system. To influence encoded vector index loading:

- Specify the level of parallel processing allowed using either the QQRVDEGREE system value, HomeRun, or the CHGQRYA command.



Skip sequential processing

Read records from a file in the order they physically reside in the file eliminating records that will not be used in the result set as they are read. This approach is preferred to a table scan as it is more efficient.

Skip sequential processing is a good alternative for the following cases.

- A majority of the records in the file will be selected
- The file contains a small number of records

Influencing skip sequential processing

- Create an encoded vector index (EVI) for each of the fields use in the WHERE clause for this file. This will improve skip sequential processing efficiency and give the iSeries better statistics about data in the file.
- See *Table scan* for information more techniques.



Table scan

Reads the entire file using the physical order of the records. This optimizes disk access as it minimizes random I/O.

If the percentage of rows selected is greater than 20%, this method is more efficient than index processing. On the other hand, if a small percentage of data is selected and the file is large, this access method is not optimal.

Optimizing table scan

- A table scan is most efficient if the optimizer can implement data space selection. This is the lowest level of selection the system can perform and can greatly minimize the CPU requirements to select or reject rows. The following situations allow the optimizer to use data space selection:
 - the fields referenced in the WHERE clause should come directly from the table
 - literal values in the WHERE clause should have the length and precision of the data in the table
 - comparisons with packed decimal or other numeric types can only take advantage of data space selection if the table was created with an SQL CREATE TABLE statement
 - character fields cannot be variable length
- Specify OPTIMIZE FOR n Rows. If a large number is specified for 'n', the query optimizer will tend to choose arrival sequence more often. If a small number is chosen for 'n', the optimizer will attempt to use a keyed access method if possible. In general it is best to choose a value for 'n' that matches the number of rows you actually expect to be returned from the SQL statement.
- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will improve table scan efficiency and give the iSeries better statistics about data in the file.
- Create a traditional binary radix index for key fields used for ORDER BY, GROUP BY, or WHERE clause conditions. This will give the iSeries an index based option to access the file

instead of a table scan. This should be done in cases where less than 20% of the rows in the file will be used in the result set or if these key fields are used often in several queries.

- Add or modify an ORDER BY clause.
- Specify a field from that has an index to bias the optimizer toward index based access.
- Specify a field from another file in the query to bias index access for that file and table scan for this file.
- Specify a field from several files to bias the iSeries to use a sort rather than index access for ordered access to the data.
- Allow I/O or CPU parallel processing. Table scan processing benefits from both I/O and CPU parallel processing.
- Change the availability of hash join processing.

6.8 Integration of iSeries Navigator Visual Explain

Users can also invoke the iSeries Navigator Visual Explain tool in the much the same manner as **Visual Explain** namely by right-clicking on the cell(s) in the “Last Operation” column and passing the SQL statements located in the cell(s) to the iSeries Navigator Visual Explain tool.

Once the iSeries Navigator Visual Explain tool is activated, the user needs to enable iSeries Navigator Visual Explain feature by clicking on "Connection" --> "Connect to Server...".

To start using iSeries Navigator Visual Explain, highlight the target SQL statement and do one of the following:

- 1) click on "VisualExplain" --> "Explain" menu option
- 2) click a button with a "Visual Explain Only" screentip
- 3) type Ctrl+E

6.9 System-wide installation impacts

HomeRun changes some settings on your IBM System i™ system at installation time that can affect your current system configuration and activity. The HomeRun modifications that may have system-wide impacts are limited to:

- Changing system values
- Ending currently active Centerfield servers, jobs, and monitors
- Configuring network attributes
- Modifying subsystem configurations
- Adding exit points

This section also details information on HomeRun's use of the following IBM System i™ features:

- TCP/IP usage
- Job scheduler usage
- Authorities on installed libraries
- CL Commands and APIs used
- Programs which adopt authority

6.9.1 System Values

The HomeRun toolset helps you handle issues like performance, security, and usage tracking which requires special system privileges. Because the product installs software that performs privileged operations, it requires certain system values that control authority and application privileges to be set to certain values. The following table shows the list of system values and a description of the HomeRun installation requirements.

QALWUSRDMN	This system value is checked at installation to ensure that the value is either *ALL or that it contains the {PROGRAM LIBRARY} library as one of the libraries that allows user domain objects. HomeRun requires this setting because it stores and directly accesses user space objects in the {PROGRAM LIBRARY} library. If the system value is not set properly for HomeRun, the installation support modifies the setting to include the {PROGRAM LIBRARY} library.
QALWOBJRST	This system value is checked at installation to ensure that the value is either *ALL, or that it at least allows system state objects (*ALWSYSSTT) and objects that adopt owner authority (*ALWPGMADP) to be restored. HomeRun requires this setting because it restores objects with these attributes into the {PROGRAM LIBRARY} and {DATA LIBRARY} library. If the system value is not set properly for HomeRun, the installation will fail.

QUSEADPAUT	This system value is checked at installation time to ensure that the system does not restrict the list of user profiles that are allowed to modify programs to use adopted authority. HomeRun requires this because depending on the system program temporary fix (PTF) level, this system value can also control users that are allowed to run programs that adopt *OWNER authority. If the system value is not set properly for HomeRun, the installation will fail.
------------	--

HomeRun references several other system values at run time to determine information about your system configuration. The following are other system values that are referenced at run time:

- QACGLVL
- QDATE
- QDAY
- QYEAR
- QMONTH
- QSRLNBR

6.9.2 Servers, Jobs, and Monitors

You need to ensure that the objects that will be replaced by the HomeRun installation and the resources that are being modified by the installation are not used or locked by other jobs on the system prior to starting the product installation. HomeRun creates and replaces objects in the *{PROGRAM LIBRARY}* and *{DATA LIBRARY}* libraries. You should ensure that there are no objects locked in these libraries prior to installing the software. However, even if you do not check these libraries before you start the installation, the HomeRun installation checks that some of the common jobs that can hold critical locks or cause other problems are always ended before letting the installation continue.

6.9.3 Database Monitor and HomeRun

Several features within HomeRun build on top of the IBM System i™ database monitor support (STRDBMON command). The HomeRun installation ends the database monitor if it is actively collecting system-wide activity. It ends the monitor to help ensure that there are no locks on files within the *{DATA LIBRARY}* library.

6.9.4 Subsystems and Work Management

The HomeRun server software handles:

- Managing and processing incoming requests from the administrative console software
- Enforcing user access policies configured by the administrative console software
- Scheduling of any activity requested by the administrative console
- Maintaining data collection information

The HomeRun server uses its own work management configuration to control how

HomeRun client jobs are started. At installation time or using the HomeRun CFGSVR IBM System i™ command, you define the subsystem that HomeRun activity should run in. By default the Centerfield server will run in its own subsystem defined by the XCSBS80 subsystem description in the program library. It is highly encouraged that you use this subsystem to isolate the Centerfield server from other system work. At installation time, an autostart job entry is put into the QUSRWRK subsystem. This autostart entry will automatically start the Centerfield subsystem and server so you do not normally have to start the server manually after an IPL. **If you use the default subsystem (XCSBS80) you can skip the rest of this section.**

The job queue is used as the starting point for starting new HomeRun client jobs. The job queue entry that is added is for the XCTCP job queue that is found in the *{PROGRAM LIBRARY}* library. If you are changing an existing configuration or installing over a previous version of a HomeRun server, the existing job queue entry is first removed and then the new entry is added to the specified subsystem. The command that HomeRun uses to add the job queue entry is similar to the following. The default subsystem is XCSBS80, and the sequence number begins at 50. If 50 is not available, the number is automatically incremented by the HomeRun configuration support until an unused sequence number is found.

```
ADDJOBQE SBSD(subsystem description) JOBQ({PROGRAM LIBRARY}/XCTCP) MAXACT(*NOMAX) SEQNBR(50)
```

A secondary job queue entry, XCAUTODBA, is used to schedule background work for HomeRun's AutoDBA feature. By default it is placed at SEQNBR(60).

The routing entry that HomeRun installs controls the routing of the HomeRun client jobs. It must be added to the same subsystem that has the XCTCP job queue entry. The routing entry uses the XCTCP routing data for comparison and calls the QCMD program. The XCTCP job class is specified as the job class for the routing entry. The command that HomeRun uses to add the routing entry is similar to the following. The default subsystem is XCSBS80, and the sequence number begins at 170. If 170 is not available, the number is automatically incremented by the HomeRun configuration support until an unused sequence number is found. The second routing entry is added for work that should be run at a lower priority level and therefore uses a different class than the rest of the server jobs.

```
ADDRTGE SBSD(subsystem description) SEQNBR(170) CMPVAL(varies by release) PGM(QSYS/QCMD) CLS({PROGRAM LIBRARY}/XCTCP)
```

```
ADDRTGE SBSD(subsystem description) SEQNBR(180) CMPVAL(varies by release) PGM(QSYS/QCMD) CLS({PROGRAM LIBRARY}/XCTCPBJ)
```

If you are changing an existing configuration or installing over a previous version of HomeRun, the existing routing entry is not removed. It should not cause any harm to your existing application environment to leave it installed in a subsystem that is no longer used by HomeRun. If you want to remove it, you need to remove it using standard work management support available on the IBM System i™. You can use a command similar to following. Before issuing the command, you need to make sure that the sequence number used on the remove routing entry command is the sequence number for the HomeRun routing entry. You can check the routing entries by using the WRKSBSD command and displaying the routing entries for the subsystem that you want to change. For more information about performing work management activities, see IBM System i™ work management documentation.

RMVRTGE SBSD(subsystem description) SEQNBR(170)

RMVRTGE SBSD(subsystem description) SEQNBR(180)

The HomeRun installation creates objects within the {PROGRAM LIBRARY} installation library to support these and other work management changes on your system. The following are objects that are created by the HomeRun installation support that can be used for work management control.

Object Name	Object Type	Description
XCTCP	Job description	Job description used to set job properties for HomeRun administration client jobs
XCTCP	Job class	Job class used to set job properties for HomeRun administration client jobs
XCTCPBJ	Job class	Jobs class used to run jobs that may run long- running operations and should execute at a lower priority than other work.
XCTCP	Job queue	Job queue used to start HomeRun administration client jobs
XCHLP	Job description	Used by the XCHELPER background job
XCIAFINDER	Job description	Used by the XCIAFINDER background job (V5R4 and higher)
XCPC	Job description	Used by the XCPC background job
XCSTRSVR	Job description	Used by the autostart job entry
XCSCRUBBER	Job description	Job description used to set job properties for the HomeRun Collection Scrubber job



{PROGRAM LIBRARY}	Output queue	Output queue used for upcoming product enhancements
----------------------	-----------------	---

You can modify the objects used for controlling work management, but your changes will not be preserved when you upgrade to the next version of HomeRun.

6.9.5 Exit points

The following exit point is added during HomeRun product installation:

Exit Point	HomeRun Program Name
QIBM_QWT_JOBNOTIFY	XCDTAQ

This exit point is used by the Usage Tracker for insure/INDEX and insure/ANALYSIS. At installation time, no jobs are ended and no subsystems are ended. However, if you want to have the Usage Tracker monitor for specific new jobs that start (that is, if you choose the *Filtered jobs* Profile type), you will need to end and restart any subsystems in which those jobs might start before your monitor will take affect. This only needs to be done once after the installation of HomeRun.

Three other exit points are optionally added at installation time. A prompt will appear that asks if these exit points should be added so that additional types of data collection can be done by HomeRun. By default they are installed, but they can be bypassed if so desired. The three exit points are:

Exit Point	HomeRun Program Name
QIBM_QSQ_CLI_CONNECT	XCCLIINIT
QIBM_QZDA_INIT	XCODBCINIT
DDM exit point (DSPNETA)	XCDDM

6.9.6 TCP/IP Usage

The HomeRun server uses TCP/IP to communicate with the HomeRun clients. The server and client applications communicate almost exclusively using a TCP application. The application is written to use proprietary application data flows to give fast and efficient performance.

The IBM System i™ and personal computer sockets applications communicate with each other through TCP ports. Ports are used by the TCP protocol to identify a unique origin or destination of communication with a TCP application. TCP ports can be any numeric value from 1 to 65535. TCP applications that are commonly used, like ftp and telnet, use pre-assigned port numbers. Pre-assigned port numbers are called well-known TCP ports.

The well-known TCP port numbers range between 1 and 1023 and should not be used when you configure HomeRun. If you specify one of these ports, it can affect the operation of the application that normally uses the well-known port. When you install HomeRun, a port and associated application service is configured. The service name for HomeRun is CENTERFIELD_SERVER_80 and the port by default is **{default port}**. If the **{default port}** port is already used by another application, the port number is incremented until a free port is found. The command that HomeRun uses to add the service and port configuration is similar to the following. You can check the port and service configuration after you install by using the WRKSRVTBLE command and locating the CENTERFIELD_SERVER_80 in the service list.

```
ADDSRVTBLE SERVICE(CENTERFIELD_SERVER_80) PORT({default port})  
PROTOCOL('tcp') TEXT('Centerfield Technology Server')
```

In addition to the proprietary communications method, HomeRun uses an ODBC connection to handle report requests made by the administration client. HomeRun requires an iSeries Navigator ODBC connection. When the HomeRun client is installed, the auto-configuration support will attempt to configure an ODBC data source if the supported ODBC driver can be detected.

6.9.7 Job Scheduler Usage

The IBM System i™ built-in job scheduler is used by several components of HomeRun. Events that can cause jobs to be placed on the job scheduler are:

- Scheduling a Database Profiler data collection
- Scheduling Index Create requests using insure/INDEX or Visual SQL Explain
- Cleaning database collections using the Collection Scrubber
- Autonomic Database Assistant (AutoDBA) analysis and actions

6.9.8 Default public authority of libraries

The *{PROGRAM LIBRARY}* and *{DATA LIBRARY}* libraries are created with the default create authority set to *CHANGE.

6.9.9 Command Language (CL) Commands Used by HomeRun

HomeRun is continually being enhanced and modified, so the list of CL commands used by HomeRun also continues to change. The following is a list of CL commands that are used by HomeRun. If you have modified command defaults or installed an application that either replaces or changes any of the following commands, you should contact Centerfield Technology to discuss the impacts that the changes may have on HomeRun.

NOTE: This list may change without notice and is not guaranteed to be complete for the most current version of software.



ADDPFTRG	CHKTAP	DLTJRN	IF	RTVMBRD
ADDMON*	CLOF	DLTJRNRCV	GOTO	RTVMSG
ADDEXITPGM	CPROBJ	DLTOVR	MONMSG	RTVNETA
ADDJOBQE	CPYF	DLTPGM	MOV OBJ	RTVOBJD
ADDJOBSCDE	CPYSPLF	DLTSP*	OPNDBF	RTVSYVAL
ADDMSGD	CRTDTAARA	DLTSPLF	OVRDBF	SAVOBJ
ADDRTGE	CRTDUPOBJ	DLTUSRSPC	OVRPRTF	SBMJOB
ADDSRVTBLE	CRTJRN	DO	PASSWORD*	SNDPGMMSG
ALCOBJ	CRTJRNRCV	DSPDBR	PGM	SNDRCVF
CALL	CRTLF	DSPFD	PRTSQLINF	STRDBG
CALLPRC	CRTLIB	DSPJOB	RETURN	STRDBMON
CFGSVR*	CRTOUTQ	DSPJOBLOG	RCVF	STRHOSTSVR
CHGACGCDE	CRTPF	ENDDBG	RCVMSG	STRJRNPF
CHGJOB	CRTSAVF	ENDDBMON	RMVJOBQE	STRSVR*
CHGLIBL	CRTSP*	ENDDO	RMVJOBSCDE	STRTCPSVR
CHGNETA	CRTSRCPF	ENDHOSTSVR	RMVMSGD	
CHGOBJOWN	CRTUSRPRF	ENDJOB	RMVPFTRG	
CHGQRYA	DCL	ENDJRNPF	RMVMON*	
CHGSYSLIBL	DCLF	ENDPGM	RNMOBJ	
CHGSYSVAL	DLCOBJ	ENDPJ	RSTOBJ	
CHGVAR	DLTDTAARA	ENDSVR*	RTVDTAARA	
CHKOBJ	DLTF	ENDTCPSVR	RTVJOBA	

* Indicates Centerfield Technology command

6.9.10 OS/400 System APIs Used by HomeRun

HomeRun is continually being enhanced and modified so the list of system application interfaces (APIs) used by HomeRun also continues to change. The following is a list of system APIs and header files that contain API interfaces that are used by HomeRun. If you have installed programs that either replace or change any of the following APIs, you should contact Centerfield Technology to discuss the impacts that the changes may have on HomeRun.

NOTE: This list may change without notice and is not guaranteed to be complete for the most current version of software. This list may not include every API that is used if the API name does not match the included source member (i.e. the name of the API is in the source file and used by Centerfield but not explicitly listed here).

CEELOCT	QUSDLTUS	QWTSETP
QCMDEXC	QUSGEN	QWCRSVAL
QDBRTVFD	QUSLJOB	QUSRTVEI

QDBLDBR	QUSLMBR	QUSMBRD
QLICHGLL	QUSLOBJ	QUSRTVFD
QMHRVTVM	QUSPTRUS	QUSCHGPA
QMHSNDPM	QUSRJOBI	QDBBRCDS
QSYGETPH	QUSRMBRD	QTNADDCR
QSYRLSPH	QUSRTVUS	QUSMIAPI
QUSCHGUS	QWCRJBST	QDMLOPNF
QTNRMVCR	QWCRSSTS	QMHRSNEM
QUSEC	QWCLOBJL	QPMWKCOL
QJOURNAL	QMHRCVPM	QWDLSJBQ
QTOCNETSTS	QPMLPFRD	QWCLSCDE
QWCCHGTN	QWDLSBSE	
QPMLPMGT	QUSCUSAT	

6.9.11 Programs Adopting *OWNER Authority

The following programs adopt *OWNER authority. All HomeRun programs are compiled to use adopted authority.

Program Name	Description
XCACTODBC	Retrieve actively connected user access jobs
XCADDJE	Add job notify exit point
XCADDJOB_CFG	Add job configuration command processor
XCADDUAE	Add the remote monitoring support
XCADVIX	Advise indexes
XCAUDOPR	Audit a client request
XCJBCFGPOP	Change job history prompt override processor
XCCFGMONHS	Configure monitor history auditing
XCCFGMONHP	Configure monitor history prompt override program
XCCHGODBCA	Change run time attributes of a user access job
XCCHKCMTCTL	Check commitment control driver program
XCCMD	Internally used by client software to run a Command Language (CL) command
XCCMT (service program)	Commitment control API interfaces
XCDSPLICINF	Display system license information
XCENDHLPCP	End helper command processing program
XCENDSCRBR	End scrubber job command processor
XCHELPER	Background job reading job notify data queue
XCHLP	Main helper job program

XCIAFINDER	Read journal to retrieve recommended indexes
XCIP	Capture IP address for job
XCJOBUTIL (service program)	Job APIs that require job control authority
XCJDELOG	Look at JDE logs
XCJDEINI	Edit JDE.INI file
XCJOBPRF	Start a database monitor collection for a job
XCLOCKCF	Lock APIs
XCLSTJOB	List jobs APIs
XCMONHST	Monitor job history to audit table
XCMTXWAIT	Mutex APIs
XCPASS	License entry and verification
XCPERFCOLL	Performance collector APIs
XCPMSUB	Dynamically substitute parameters for a parameterized SQL SELECT statement
XCPRFCRT	Create a database collection profile
XCPRFDTL	Delete a database collection profile
XCPRFEND	End a Database Profiler collection
XCPRFENDAP	End all active Database Profiler collections
XCPRFRNM	Rename a database collection profile
XCPRFSTR	Start a Database Profiler collection
XCOWCFG	Configure server for OneWorld environment
XCREPBLD	Build Database Profiler repository
XCREPDLT	Delete repository
XCRTVCFGVAL	Retrieve configuration value
XCRMVJBCFG	Remove job configuration
XCRMVUAE	Remove the remote monitoring support
XCSCDTRG	Schedule requested work (used as trigger program for XCSCD file)
XCSCRUB	Remove unneeded collection data
XCSEMWAIT	Semaphore APIs
XCSIGNON	Check password at sign on time and swap user profile to signed in user
XCSPDSPCMD	Internally used by client software to run an OS/400 command and return spool file output
XCSQLEXEC	Internally used by client software to run non-SELECT SQL statement with commitment control
XCSQLEXEC0	Internally used by client software to run non-SELECT SQL statement without commitment control

XCSQLSEL	Internally used by client software to run SQL SELECT statement with system naming
XCSQLSEL2	Internally used by client software to run SQL SELECT statement with SQL naming
XCSTRHLPCP	Start helper command processing program
XCSTRSVR	Start the Centerfield server
XCSYSUTIL (service program)	System information APIs
XCTRIGGER	Trigger program for configuration files
XCTRIGGERM	Trigger program for job monitor configuration files
XCDBGUTIL (service program)	Debug utilities

6.10 Enabling remote monitoring

To enable the remote monitoring functionality of some of the HomeRun tools, the ADDMON command must be run on your IBM System i™ server. You should enable remote monitoring if you want to:

1. use insure/MONITOR to monitor and track usage by users of remote interfaces, such as ODBC or FTP
2. use insure/SECURITY to prevent access by unauthorized users of remote interfaces.

6.10.1 Effects of exit point registration on your environment

The ADDMON command can be found in the HomeRun server installation library. The command allows you to specify the interfaces that you want to monitor and control. See your documentation for insure/MONITOR or insure/SECURITY for a list of supported interfaces.

When the ADDMON command is run, some control information is created within the installation library, and exit programs for the requested interfaces are registered on the system. For all registration facility exit points used by HomeRun, any currently configured exit programs will be replaced with HomeRun's exit programs. This is done because the operating system only honors one exit program per exit point for several of the available exit points. If an exit program was previously configured, the installation support saves the reference to the exit program in an internal area. When the HomeRun exit program is called, HomeRun's security, resource, and auditing policies will be enforced. If a user exit program was configured before HomeRun was installed, and HomeRun's policies did not reject the connection, the previously defined user exit program will be called *in addition* to the HomeRun programs, and its output will be returned as if it had been called directly.

If the DDM access exit point had been configured to a value other than the default, it is handled in a similar manner. HomeRun's exit program is called and the configured HomeRun policies are enforced. If a non-default value was detected for the DDM access exit point at HomeRun installation time, and HomeRun did not reject the connection, either the user defined exit program will be called or the previously defined action will be taken. The output returned from the exit program will be functionally consistent to what would have been returned had HomeRun not been installed.

In some cases you may want HomeRun's exit program called *after* your existing exit program. You may or may not be able to make this work depending on the design of your existing exit program and the method that you use to install your program.

- If your exit program is registered by installing a third party application, you need to

ensure that the third party application has a method for remembering and calling exit programs that exist on an exit point prior to their installation. You should also ensure that when the third party application is removed that the application restores the exit point to its original state.

- If your exit program is a custom program designed by your own staff and installed directly using operating system commands, you need to ensure that you have designed it to have a method for knowing about and calling HomeRun's exit program after it has done its processing. In both cases, the HomeRun exit program must be passed the same parameters that it would have been passed if the operating system had called the program directly. If all of your exit programs meet these criteria, you can achieve any exit program call order that you would like. The call order will be a Last In First Out (LIFO) ordering scheme. The last exit program installed will be the first exit program called when the exit point is encountered. To force HomeRun's exit program to be called last you need to:
 1. De-register all of the user exit programs that you currently have configured for each exit point that the HomeRun server uses. See the next section to determine the specific exit points used by HomeRun. If you do not know whether you use exit programs today, you can use the WRKREGINF and DSPNETA commands to display the currently registered exit programs for the various exit points.
 2. Run the ADDMON command to register the HomeRun exit programs and enable the remote monitoring support.
 3. Re-register each of your user exit programs. If you have multiple exit programs to install on the same exit point, you should record the order that you use to install them to ensure that if you need to remove them that you use exactly the reverse order.

6.10.2 Exit Program Registration Details

HomeRun registers exit programs using the IBM System i™ registration facility support for several products and features. You control the exit programs that are installed by specifying parameters on the ADDMON command.

The exit programs that are registered by HomeRun enforce the HomeRun policies. The installation of an exit program can affect your current system activity. Most of the OS/400 products and features that allow exit programs to be registered do not fully reinitialize when an exit program is installed. The issues surrounding how exit programs are added, resolved, and called can cause run time failures if an exit program is replaced after a job has initialized the calling information for an exit program. ***For this reason, any system activity for the products and features that have exit points that are modified by the HomeRun support should be ended prior to running the command, and not resumed until the command completes.*** If any job using these products or features remains active or becomes active through the course of the installation, unexpected run time errors may occur the next time that the job calls the registered exit program. ***Note:***

IBM documentation states that if the exit program for File Server (QIBM_QPWFS_FILE_SERV) is changed, the QSERVER subsystem must be ended and restarted for the changed to take effect.

To manually end OS/400 products and features that have exit points that will be modified by HomeRun, you can use the following commands and techniques:

- To end FTP:
QSYS/ENDTCPSVR SERVER(*FTP)
- To end IBM ODBC, first issue this command:
QSYS/ENDHOSTSVR SERVER(*DATABASE)

Then do one or both of the following:

If your ODBC users connect over APPN use this command:

QSYS/ENDPJ SBS(QSERVER) PGM(QIWS/QZDAINIT)

If your ODBC users connect over TCP/IP use these commands:

QSYS/ENDPJ SBS(QSERVER) PGM(QIWS/QZDASOINIT)

QSYS/ENDPJ SBS(QSERVER) PGM(QIWS/QZDASSINIT)

To restart the servers after the exit points are installed, issue the following commands:

- STRHOSTSVR SERVER(*DATABASE)
- STRTCPSVR SERVER(*FTP)

Exit points that the ADDMON command modifies are specified in the following table.

Exit Point	ADDMON option specified
QIBM_QZSC_SM	*CSCM
QIBM_QZSC_NLS	*CSCONV
QIBM_QZSC_LM	*CSLM
QIBM_QZHQ_DATA_QUEUE	*DTAQ
QIBM_QNPS_ENTRY	*NETPRT
QIBM_QVP_PRINTERS	*VRTPRT
QIBM_QHQ_DTAQ	*DTAQORIG
QIBM_QTF_TRANSFER	*FILETRANS
QIBM_QLZP_LICENSE	*LMORIG
QIBM_QMF_MESSAGE	*MSGORIG
QIBM_QRQ_SQL	*RMTSQL
QIBM_QZRC_RMT	*RMTCALL
QIBM_QZRC_RMT	*RMTCMD

QIBM_QPWFS_FILE_SERV	*FILSRV
QIBM_QTMX_SVR_LOGON	*REXEC
QIBM_QZSO_SIGNONSRV	*SOCKSRV
QIBM_QTG_DEVINIT	*TELNET
QIBM_QZDA_SQL1, QIBM_QZDA_SQL2	*IBMODBC
QIBM_QTMF_SERVER_REQ, QIBM_QTMF_CLIENT_REQ	*IBMFTP
QIBM_QZDA_INIT	*ODBCINIT
QIBM_QSQ_CLI_CONNECT	*CLIINIT
QIBM_QZDA_INIT QIBM_QSQ_CLI_CONNECT DDMACC in CHGNETA command	*INSURE_IA

Some data access interfaces like DRDA based ODBC do not support the registration facility. To support DRDA based data access, HomeRun registers the XCDDM exit program under the network attributes DDM access exit point. On V4R1 and above, the IBM System i™ DRDA support calls the DDM access exit program when a connection is established. Prior to V4R1, the exit program is not honored for DRDA. If the DDM access setting is a value other than *OBJAUT prior to installing the E-Connect Server, the installation support will honor the previously configured value in addition to the E-Connect Server's configured policies.

Interfaces that are controlled via the DDM access exit point are shown in the following table. To specify these interfaces on the ADDMON command, use the *IBMDDM parameter.

Distributed Data Management
DRDA - FileTek products
DRDA - Grandview DB/DC Systems products
DRDA - IBM DB2 Connect (formally DDCCS)
DRDA - IBM DB2 for VSE and VM
DRDA - IBM DB2 UDB for IBM System i™
DRDA - IBM DB2 UDB for OS/390
DRDA - Informix Software products
DRDA - Oracle Corporation products
DRDA - StarQuest products
DRDA - Wall Data Rumba for Database Access
DRDA - XDB Systems products
DRDA - Derby Network Client
DRDA - Java Client (JCC)
DRDA - DataDirect Technologies
DRDA - SeeBeyond ICAN (Sun JCAPS)

If manual steps for recycling servers in question are a hassle, there is an 'automated' alternative. At the end of "HomeRun Installation Guide" document there is a section titled "Recycle host servers sample CLLE script" which outlines steps to create a

RECYCLESVR program which can then be invoked to recycle the host servers upon running of RMVMON command. Sample code refers to default installation settings.

6.11 Removing remote monitoring support

The HomeRun remote monitoring support can be disabled by running the RMVMON command that can be found in the HomeRun installation library. The command allows you to disable the support for interfaces that you specify. The default value is to remove all interfaces.

If you registered additional exit programs on the same exit points used by HomeRun after the HomeRun server was installed, you need to ensure that you de-register the additional exit programs *before* disabling the HomeRun remote monitoring support. If any registration facility user exit programs were replaced when the HomeRun remote monitoring support was added, they will be restored to their original values. Likewise if the DDM access exit point had been configured to a value other than the default prior to the registration of HomeRun remote monitoring support, it will be restored to its previous value.