



@server

iSeries

Administering iSeries Access for Windows





@server

iSeries

Administering iSeries Access for Windows

Contents

| | |
|--|----|
| Administering iSeries Access for Windows | 1 |
| What's new for V5R2 | 2 |
| Print this topic | 3 |
| iSeries Access for Windows network environments. | 3 |
| Microsoft Windows Terminal Server | 3 |
| Using iSeries Access for Windows in a three-tier environment | 4 |
| Using Microsoft Transaction Server (MTS). | 5 |
| Accessing iSeries services from the middle tier | 5 |
| Add TCP/IP configuration to all users | 7 |
| Set PC5250 files location for all users | 7 |
| User profiles for PCs with multiple users | 7 |
| Installing or migrating on multiple PCs | 8 |
| Creating a tailored installation image of iSeries Access for Windows | 9 |
| Performing a silent installation of iSeries Access for Windows | 9 |
| Creating response files for iSeries Access for Windows installations | 11 |
| Starting a silent installation | 11 |
| Return codes for silent installations or migrations | 12 |
| Administering service packs | 12 |
| Check service level. | 13 |
| Installing the service pack silently | 13 |
| ODBC administration | 14 |
| Overview of the iSeries Access ODBC driver | 14 |
| Setting up your system for the iSeries Access ODBC driver | 15 |
| Adding the local system to the RDB directory | 16 |
| Specify the ODBC data source | 17 |
| iSeries Access for Windows ODBC security | 17 |
| Risky ODBC security strategies | 17 |
| ODBC program security strategies | 18 |
| Related information for ODBC security. | 19 |
| Troubleshoot ODBC | 19 |
| ODBC diagnostic and performance tools | 20 |
| iSeries Access ODBC error messages | 21 |
| Troubleshooting the iSeries server connection | 22 |
| Common ODBC errors | 24 |
| Gathering information for IBM Support. | 26 |
| Host server administration | 26 |
| OS/400 host servers | 27 |
| Host servers by iSeries Access for Windows function | 28 |
| File Server | 29 |
| Database server | 29 |
| Data Queue Server. | 34 |
| Network Print Server | 34 |
| Central Server | 34 |
| Remote Command and Distributed Program Call Server | 35 |
| Signon Server. | 35 |
| Server Port Mapper. | 35 |
| Using OS/400 host servers | 36 |
| Establishing client/server communications | 36 |
| Subsystems on the iSeries server | 40 |
| System Values on the iSeries server | 51 |
| Identifying Server Jobs on the iSeries server | 54 |
| Using EZ-Setup and iSeries Navigator with host servers | 56 |
| Using server exit programs | 56 |

| | |
|---|-----|
| Registering Exit Programs | 56 |
| Writing Exit Programs | 59 |
| Exit program parameters | 60 |
| Examples: Exit Programs | 80 |
| Integrating new functions into iSeries Access for Windows and iSeries Navigator | 94 |
| Integrating Plug-ins | 94 |
| Integrating add-ins | 95 |
| iSeries NetServer administration | 96 |
| Restricting users with policies and application administration | 97 |
| Overview of iSeries Access for Windows policies | 97 |
| Types and scopes of policies | 98 |
| Setting up your system to use policies | 99 |
| Configuring an iSeries server for policies | 99 |
| Configuring client PCs for policies | 99 |
| Creating policy files | 100 |
| iSeries Access for Windows policy list | 101 |
| Policies by function | 102 |
| Policies by template | 104 |
| Secure Sockets Layer administration | 105 |

Administering iSeries Access for Windows

This topic assumes that you are already familiar with iSeries Access for Windows, and have installed it on your system. For an overview of iSeries Access for Windows and a description of how you can use it in your network, refer to the Get Started topic. For help with installing and setting up iSeries Access for

Windows, refer to iSeries Access for Windows - Setup .

This topic can help you with administration issues regarding iSeries Access for Windows.

iSeries Access for Windows network environments

Learn about some of the network environments in which iSeries Access for Windows can operate. In particular, learn how to make OS/400 services available to your clients by using iSeries Access for Windows in a three-tier environment, or by installing it on Windows NT Server 4.0 Terminal Server Edition or on Windows 2000 using Terminal Services. Also, learn how to administer a PC that will have multiple users assigned to it.

Installing or migrating on multiple PCs

You can install iSeries Access for Windows on multiple PCs and specify the components you want, without going through the steps of the initial installation and configuration.

Administering service packs

Learn about PTFs and service packs, and how to use the Check Service Level function to administer them.

ODBC administration

iSeries Access for Windows includes an ODBC driver that can allow your applications convenient access to DB2 UDB for iSeries databases in your network. This topic provides an overview of ODBC, instructions for setting up the driver, and a troubleshooting guide.

For information about using and implementing the ODBC APIs, refer to ODBC programming.

Host server administration

This topic describes the host servers that are commonly used with iSeries Access for Windows, and describes how to effectively manage and use them.


Integrating new functions into iSeries Access for Windows and iSeries Navigator

You can extend iSeries Access for Windows and iSeries Navigator functions using customized, or third party, applications called plug-ins and add-ins. Learn how to integrate these programs into your system, and then use iSeries Access for Windows to distribute and maintain them.

Setting restrictions using policies and Application Administration

iSeries Access for Windows provides multiple methods of setting up restrictions and profiles. These include policies that can be set using Microsoft's policy editor, and the Application Administration function of iSeries Navigator.

Administering iSeries Access for Windows requires a knowledge of a number of related topics as well. You may need information on the following topics:

- Secure Sockets Layer (SSL)
- AS/400 NetServer
- iSeries Access for Windows on Windows 2000 implementation notes 
- Programming for iSeries Access for Windows

There are many tools available that will track all of the changes made to a PC by an installation program. At the time of the publication, several of the tools we found were available for download from ZDNet and InstallSite on the General tools > Analyzing a setup page. These tools and web sites are in no way affiliated with IBM.

Note: Read the Code example disclaimer for important legal information.

What's new for V5R2

New features for iSeries Access for Windows administrators, include:

- **64-bit ODBC/OLE DB support**
iSeries Access for Windows now provides support for both a 32-bit and 64-bit ODBC driver. The 64-bit ODBC driver is automatically installed along with the 32-bit ODBC driver when running under a 64-bit version of Windows. ODBC applications running in 64-bit versions of Windows will automatically use the appropriate ODBC driver, depending on what bit version the application was compiled for. For example, the 64-bit driver can only be used by a 64-bit application. For more information, see 64-bit ODBC Support, in the iSeries Access for Windows' User's Guide.
- **Silent Install Indicator**
New for V5R2, silent install now has a progress indicator. The Silent Install Indicator will be an icon in the task tray which will appear when a silent install is launched, and remain in the task tray as long as the install is executing. The Silent Install Indicator can be expanded to expose information regarding the install. For more information, see Performing a silent installation of iSeries Access for Windows.
- **Kerberos support**
iSeries Access for Windows now supports using a Kerberos principal name instead of a userid and password to authenticate a user when connecting to an iSeries server. This option is available when connecting from Microsoft Windows 2000, XP, and later operating systems that support the Kerberos protocol. iSeries servers that are V5R2 or later, can be configured to participate in a Kerberos-enabled network through Network Authentication Service. For more information, see Network authentication service.
- **Enhanced CWBCFG PC command**
The CWBCFG PC command has been enhanced to allow setting the location where the PC5250 emulator looks for and stores files, for all users of a PC. For more information, see Set PC5250 files location for all users.
- **Independent ASP support**
iSeries Access for Windows now supports accessing multiple databases through Independent ASPs. For more information, see Setting up your system for the iSeries Access ODBC driver .
- **Tailored install can include SSL**
If SSL support is installed on the image you are using to create your tailored installation image, the SSL support can be included in the tailored image. For more information, see Creating a tailored installation image if iSeries Access for Windows.
- **iSeries ODBC Driver for Linux**
You can install Linux on an iSeries logical partition and use the iSeries ODBC Driver for Linux to access the iSeries database.

Note: iSeries ODBC Driver for Linux is not part of iSeries Access for Windows. It is a separate product used only with the Linux operating system.

- **Name changes**
 - The Client Access Express Remote Command service is now called the iSeries Access for Windows Remote Command service.
 - The Client Access ODBC driver (32-bit) is now called the iSeries Access ODBC driver. Note: Client Access ODBC driver (32-bit) will remain for compatibility purposes, but data sources should be migrated to the new name. Both names refer to the same driver.
- **No longer available**

- 56-bit SSL encryption (CE2) is no longer available. Only 128-bit SSL encryption will be supported.
- The Windows 95 operating system is not supported with V5R2 iSeries Access for Windows.

Print this topic


To view or download the PDF version, select iSeries Access for Windows Administration (about 350 kb, or 114 pages).

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As...**
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

iSeries Access for Windows network environments

iSeries Access for Windows provides several methods of providing end users with access to iSeries services. Typically this involves a direct connection between a PC running iSeries Access for Windows and the iSeries server. However, the following methods allow you to take advantage of other networking environments.

- **Microsoft Windows NT 4.0 Terminal Server Edition (TSE) or Windows 2000 running terminal services**

TSE is a multi-user version of Windows NT server 4.0 that allows multiple, simultaneous client sessions to run on a single NT 4.0 server. TSE allows connections from multiple platforms, including network stations, UNIX, DOS, OS/2 and many other types of workstations. By installing iSeries Access for Windows on the TSE server, you can provide access to iSeries services from workstations that do not have iSeries Access for Windows installed. These functions are also available with Terminal Services, a feature in all server versions of Windows 2000.

- **iSeries Access for Windows in a three-tier environment**

By installing iSeries Access for Windows on the middle-tier of a three-tier environment, you can provide a wide variety of client workstations access to iSeries services. Additionally, three-tier environments present several other advantages, like enhanced transaction management.

iSeries Access for Windows also provides ways to administer PCs with multiple users:

- **Add TCP/IP configuration to all users**

Use the CWBCFG command to configure iSeries server connections for all users on a Windows NT/2000/XP workstation or server.

- **User profiles for PCs with multiple users**


Windows operating systems allow you to use roving, roaming and mandatory user profiles to manage PCs that have more than one user.


Microsoft Windows Terminal Server

Microsoft Windows Terminal Server Edition (TSE) is a multi-user version of Windows NT Server 4.0 that allows multiple, simultaneous client sessions to run on a single NT 4.0 server. TSE allows connections from multiple platforms, including network stations, UNIX, DOS, OS/2 and many other types of

workstations. By installing iSeries Access for Windows on the TSE server, workstations that do not have iSeries Access for Windows installed can access iSeries services. These functions are also available with Terminal Services in all server versions of Windows 2000.

Note: Set **When to check service level** to **Never** on the **Service** tab of iSeries Access for Windows Properties when using Windows 2000 running Terminal Services.

For information on install, support, known problems, and solutions when using iSeries Access for Windows with a Microsoft Windows Terminal Server, refer to APAR II11373 

For more information about TSE in general, refer to the Microsoft Windows NT Server 4.0 Terminal Server Edition web site. 

Using iSeries Access for Windows in a three-tier environment

By installing iSeries Access for Windows on the middle tier of a three-tier environment, a wide variety of client workstations can access iSeries services. Additionally, three-tier environments present several other advantages:

- **Improved integration between diverse clients and server applications:** Multiple end-user applications running on various clients may communicate with multiple applications on a Windows NT/2000 server simultaneously. Each of the applications on the Windows NT/2000 server may also be communicating with multiple databases.
- **Enhanced transaction management using Microsoft Transaction Server (MTS):** Three-tier environments allow for more complex transactions, some of which may depend upon each other for their own successful completion. (All transactions must complete successfully in order for any of them to complete.)
- **Importing data from an iSeries server into web pages, using Microsoft Internet Information Server (IIS):** IIS can use Active Server Pages to dynamically update web pages with data from a DB2 Universal Database for iSeries.

All three-tier environments separate components and applications into three layers. The three layers may reside on separate PCs, or terminals, and communicate over a network. Generally the tiers will have the following characteristics:

Client tier

This layer contains the interface and applications that allow end users to manipulate data. For example, this may involve a web browser running on a network station, or a custom-built application using a remote component. This layer does not use the iSeries Access for Windows client.

Middle tier

This layer contains the business or application logic. In environments using iSeries Access for Windows, this layer should consist of a Windows server running a Microsoft Active Server Pages script or a remote component. Additionally, this layer uses Microsoft's Internet Information Server (IIS) and Microsoft Transaction Server (MTS) to manage transactions with the client tier. iSeries Access for Windows uses the ODBC driver to support MTS on the clients, and handles communication with the database tier. Microsoft currently recommends using OLE DB, ActiveX Data Objects (ADO) and Remote Data Service to access data from a component on the middle tier.

Refer to the following topics for more information about the middle-tier:

- MTS
- Access iSeries services from the middle-tier

Database tier

This layer usually consists of a DB2 Universal Database for iSeries database. Your applications can access this and various iSeries services through host server programs, or through custom-built iSeries programs.

Using Microsoft Transaction Server (MTS)

The iSeries Access for Windows client supports MTS version 2.x, and later, with the iSeries Access ODBC driver, for V5R1 or later servers.

MTS

MTS is a Microsoft component-based programming model and run-time environment for developing, deploying, and managing Internet server applications. In many three-tier environments, Active Server Pages (ASP) call MTS components to access databases, mainframe applications, and message queues. Used with iSeries Access for Windows running in the middle-tier of a three-tier environment, MTS components manage transactions between client applications, iSeries Access for Windows components, and the databases involved in the transactions.

MTS uses Microsoft Distributed Transaction Coordinator (MSDTC) in order to manage transactions that span multiple Database Management Systems (DBMS), and to ensure two-phase commit integrity when dealing with transactions whose implementations depend on mutual success.

Implementation notes

- If the MSDTC cannot load the iSeries Access ODBC driver, the `SQLSetConnectAttr(SQL_ATTR_ENLIST_IN_DTC)` will fail with reason code of 2 (XaRmCreate failed). If you installed PC5250, the MSDTC system environment path is set for you. To avoid this, the system environment path on the PC running MSDTC must include the path to the Shared directory within the directory in which iSeries Access for Windows is installed. For example: `C:\Program Files\IBM\Client Access\Shared`.
- If you are using SSL, or any other configurable value on the **Connections** → **Properties** dialog in iSeries Navigator, your iSeries connection name in iSeries Navigator must match the connection name specified on the client PC managed by MTS. MSDTC uses the same connection names as iSeries Access for Windows ODBC client PCs managed by MTS to connect to the DB2 UDB for iSeries database. To change the connection properties of the MSDTC connections, you must change the system account registry.

One way to do this is to use Incoming Remote Command (IRC) in combination with the CWBENV utility:

1. Run CWBENV on a client PC to extract the configuration information for an environment.
2. Copy the resulting file to the MSDTC PC.
3. Start the iSeries Access for Windows Remote Command service and ensure that it is configured to run in the Local System context.
4. Using the RUNRMTCMD command from a PC5250 session, send a CWBENV command to the MSDTC PC to import the environment.

See the User's Guide in the iSeries Access for Windows program group for more information on these functions.

For more information about MTS, refer to the Microsoft MTS web site. 

Accessing iSeries services from the middle tier

There are several ways to provide your middle-tier components with access to the iSeries server.

Note: Middle-tier components cannot have a user interface; therefore, if iSeries Access prompts for sign-on information, your three-tier applications may appear to hang. To prevent this, developers must use a new system object to specify required connection information (user ID and password) to the iSeries server. The prompt mode value for this object must be **prompt never**.

iSeries Access for Windows OLE DB provider

Most applications and components use the iSeries Access for Windows OLE DB provider through ActiveX Data Objects (ADO). Here are the four primary benefits to implementing this technique:

- It allows your developers to make only minor modifications to a single interface and programming technique in order to access iSeries programs, commands, SQL queries, stored procedures, and physical and logical files.
- It supports automatic data conversions between iSeries and PC data types.
- It allows you to avoid the overhead associated with SQL by providing support for record-level file access.
- It is relatively easy to implement and to develop applications. This method is generally the most simple technology for developing three-tier applications.

See OLE DB programming for more information.

iSeries Access for Windows ODBC driver

Additionally, you can access the iSeries Access ODBC driver through either ADO or Remote Data Services (RDS), by using the Microsoft OLE DB provider for ODBC (MSDASQL). The iSeries Access ODBC driver offers two key advantages over the iSeries Access for Windows OLE DB provider:

- Greater SQL functionality
If you need updateable cursors, SQL commitment control, or stored procedure multiple result set, consider using the ODBC driver.
- Connection pooling
In most MTS and ASP applications, each client request must connect and disconnect with the iSeries server. With connection pooling, the ODBC driver manager maintains a pool of persistent connections. Since the overhead required for the iSeries job startup is often greater than the request itself, this can offer a tremendous performance advantage.

For more information about accessing ODBC through ADO, see Choosing an interface to access the ODBC driver.

For other iSeries Access ODBC driver information, see ODBC programming.

Note: The iSeries Access for Windows OLE DB provider, and several functions in the iSeries Access ODBC driver, require MDAC version 2.5 or later. Refer to the MDAC requirements note for more information.

ActiveX automation objects

The iSeries Access for Windows client provides a library of new, enhanced ActiveX automation objects that your developers can use for middle-tier development. These objects provide access to:

- iSeries data queues
- Remote commands and distributed program calls
- Administration objects
- iSeries system objects
- Data Transfer access to iSeries database tables

In some cases, ActiveX objects provide greater versatility and functionality than ADO, but require slightly more complex programming.

Note: The iSeries Access for Windows client includes the automation library from the Windows 95/NT client (the XD1 product). These automation objects, including database, do not support use in a three-tier environment.

Express C/C++ APIs

iSeries Access for Windows APIs provide fast, low-level access to OS/400 host servers. However, using these APIs requires developers who are experienced with C/C++. Specifically, developers must be familiar with C APIs and data types, and must also account for thread-safety considerations when creating their components.

Add TCP/IP configuration to all users

Use the CWBCFG command from a DOS prompt on Windows NT/2000/XP to configure iSeries server connections for all users defined on a Windows NT/2000/XP Workstation or Server. This also adds configuration information for the Windows default user, the default profile that is used for defining new users on Windows NT/2000/XP.

For more information on CWBCFG, see the online iSeries Access for Windows User's Guide.

Set PC5250 files location for all users

To set the location where the PC5250 emulator will look for and store files for all defined users, use the CWBCFG command from a DOS prompt in Windows NT/2000/XP. If CWBCFG is never run to set this location, it defaults to (iSeries Access for Windows install folder)\emulator\private, which is shared by all users of the PC, but may not be writable by all users.

Because CWBCFG applies the setting to the Windows default user, any user account created after CWBCFG is run, will use the location set by CWBCFG rather than the normal default listed above.

For more information about CWBCFG, see the online iSeries Access for Windows User's Guide.

User profiles for PCs with multiple users

You can administer PCs with multiple iSeries Access for Windows users. This type of administration is available as a function of the Windows operating systems through the use of roving, roaming, and mandatory profiles.

Note: For documentation on how to implement these methods of multiple user administration in your network, see the Microsoft Resource Kit for the Windows operating system you are using. Resource kits are available from Microsoft, and are included with the Microsoft Developers Kit.

Roving user profiles

Roving user profiles are Windows 95/98/Me user profiles that can move between PCs that are running those operating systems. Information like desktop settings, start menu choices, and the registry reside in the user's home directory on a file server. The roving user profiles can only move between Windows 95/98/Me PCs.

Roaming user profiles

The roaming user profiles are Windows NT/2000/XP user profiles that can roam between PCs. The configuration changes go with the user. The roaming user profiles generally reside on an NT/2000/XP server. Each roaming user has a directory on the NT/2000/XP server specified by the user profile path in

the user profile settings. This directory contains registry information as well as start menu and desktop information for each user. The roaming user profiles can only roam between Windows NT/2000/XP PCs.

Mandatory user profiles

Mandatory user profiles are user profiles that a system administrator sets up for use by PC users on any Windows PC. These users typically should not modify their settings. Mandatory user profiles can exist on one PC or roam between PCs.

Installing or migrating on multiple PCs

There are several ways to install iSeries Access for Windows on multiple PCs without going through all of the steps of your initial installation and setup. Additionally, you can restrict users' access to functions by selecting which components to include in an installation.

Considerations

- **MDAC Requirements**

iSeries Access for Windows does not install MDAC like it did in previous releases. Be aware that the iSeries Access ODBC driver and iSeries Access for Windows OLE DB provider do, however, have some specific requirements on the MDAC level on your PC. Windows 98/NT/Me users should make sure the required MDAC level is on the PC before installing iSeries Access for Windows. Windows 2000 and later operating systems already have the required MDAC level.

Required MDAC levels:

- iSeries Access ODBC driver - MDAC 2.5 or later for connection pooling and MTS support
- iSeries Access OLE DB provider - MDAC 2.5 for all functions

If MDAC 2.5 or later is not installed, iSeries Access for Windows will not allow the OLE DB component to be installed. If you have a Typical install with a previous version of iSeries Access for Windows, and then you want to upgrade to V5R2M0, the OLE DB component will be deleted from your PC if MDAC 2.5 is not installed before the upgrade. You can download MDAC 2.5 or later from this Microsoft Web

Site: <http://www.microsoft.com/data>  .

- **Migration Support**

iSeries Access for Windows only supports the migration of information from:

- Client Access Enhanced for Windows 3.1 (XK1)
- Client Access for Windows 95/NT (XD1), V3R2M0

Several common installation methods are:

- **Creating a tailored installation image**



You can create a tailored installation image by excluding the unwanted components from a master installation image. You can then use the tailored installation image for installations across your network.

- **Installing or migrating silently**

Create a response file that contains a record of your responses to prompts during an installation. You can then use this response file to control duplicate installations that do not require any user interaction.

Not all of the necessary installation files reside in the same directory. To find the required files, iSeries Access for Windows searches the subfolders of the ProdData directory. See path discovery for more information.

There are many tools available that track all of the changes made to a PC by an installation program. At

the time of publication, several are available for download from ZDNet  and InstallSite  on the **General tools** → **Analyzing a setup** page. These tools and Web sites are not affiliated with IBM.

Creating a tailored installation image of iSeries Access for Windows

You may want to control which iSeries Access for Windows components your users can install. One way to do this is by excluding selected components from an installation image, and then distributing this tailored installation image to your users. The Tailored Installation Image wizard provides a simple interface for this function.

Starting the Tailored Installation Image wizard

You can start the tailored installation wizard from the iSeries Setup and Operations CD, or by navigating to the installation image directory, \QIBM\ProdData\Access\Windows\Install\Image, and entering cwbinimg.

Servicing the installation image

Any tailored installation images are not updated when Program Temporary Fixes (PTFs) are applied to or removed from the iSeries server. You must re-create the installation image to get service pack updates. Alternatively, you can combine the service pack directly with your existing tailored installation image. For instructions, go to the iSeries Access website at <http://www-1.ibm.com/servers/eserver/iseries/access/>



. Click on the latest service pack and open the subcomps/ folder. Read the instructions in the Readme.1st file located in this folder.

Distributing the installation image

The wizard allows you to specify where you want to create the tailored installation image. This location must be an empty directory, (you cannot overwrite a previous installation image) and must not be the root directory. Also, only complete installation images contain the program that creates tailored installation images. The wizard is not copied onto the user's PCs. You can also copy the tailored image to a CD-ROM. iSeries Access for Windows Setup will run automatically when the CD-ROM is inserted into the CD-ROM drive.

Note: If your iSeries server has multiple iSeries Access for Windows secondary languages, you can use any of the installed secondary languages, or the primary language on the iSeries server, as the primary language for the new installation image. This is not available if you are running the wizard from the CD, because the CD will not contain any secondary languages.

Including Secure Sockets Layer (SSL) on the installation image

If SSL support is installed on the image you are using to create your tailored installation image, the SSL support can be included in the tailored image. If tailored install detects that the SSL product is available, SSL will be displayed in the Component Selection List. SSL will not be included in the tailored image unless selected.

Note: SSL is controlled by US Export regulations. You are responsible to ensure that the new installation image is properly controlled to meet the US Export regulations.

Performing a silent installation of iSeries Access for Windows

Silent installation eliminates the need for any user interaction during the iSeries Access for Windows set up process. A response file provides all installation information so that no dialog boxes display while installing iSeries Access for Windows. To perform a silent installation:

1. Create your response file.
2. Start the silent installation.
3. Check the log file return codes to see if your installation was successful.

Note: Although Silent migrations use the same procedure as silent installations, they use a different procedure for creating the response file.

The response file contains the installation options that the system would normally prompt you for during the installation process.

Silent Install Indicator

Silent install has a progress indicator. The Silent Install Indicator is an icon in the task tray which will appear when a silent install is launched, and remain in the task tray as long as the install is executing. Passing the mouse over the icon will cause the Silent Install Indicator to display the percent of the install that is complete. The Silent Install Indicator can also be expanded to expose more information. When the install completes successfully, the icon will disappear from the task tray. If the install were to fail, the icon will remain and a small red triangle will appear on the icon to indicate the failure. Click on the red triangle to see the failure message.

Notes:

- If the Silent Install Indicator displays a given percentage of completion longer than you would expect, you may want to check the log file for errors.
- Often the best way to debug a silent install failure is to start the install in non-silent mode on the user PC, and see if there are any unexpected dialogs that appear prior to the Component Confirmation panel. Most silent install failures occur due to unexpected dialogs that appear prior to actual component installation file transfers.

Differences between normal and silent installations

The following table illustrates the differences between a normal and silent installation by comparing how the two types of installations handle various conditions that commonly arise during the installation process.

| Condition | During a normal installation... | During silent and recorded installations... |
|---|--|--|
| Select to install 5250 Display and Printer Emulator or Operations Console on Windows 95 or Windows 98. | You have the choice whether or not to write the emulator path to the autoexec.bat file. | The emulator path is automatically written to the autoexec.bat. |
| The PC has Client Access for Windows 95/NT Lightning SDK installed and, during the iSeries Access for Windows installation, you select to install Visual Basic Wizards. | A dialog box displays, warning you that Client Access for Windows 95/NT Lightning SDK will be uninstalled if Visual Basic Wizards are installed. | No dialog displays, and Client Access for Windows for Windows 95/NT Lightning SDK is automatically uninstalled while Visual Basic Wizards is installed. |
| Attempt to install a component that is restricted (by policies, dependencies, or some other restriction), or that is incompatible with a product that is already installed. | A dialog displays listing all of the components that are restricted due to these conditions. The component is not installed. | The component is not installed. |
| Perform a silent migration installation from Client Access for Windows 95/NT. | The installation directory defaults to the path where Client Access for Windows 95/NT is installed. You can change the installation directory to something other than the default, but a warning message will appear notifying you that some migrated configuration information may not work properly if you install to a new directory. | The path to install iSeries Access for Windows again defaults to the path where Client Access for Windows 95/NT is installed no matter what path you specify in the response file. |

| Condition | During a normal installation... | During silent and recorded installations... |
|------------------|---------------------------------|--|
| An error occurs. | Error messages display. | Error messages display during a recorded installation, but not during a silent installation. A negative number is written to the silent installation log file. This indicates that an error occurred. If you are having problems running silent installations, you may want to try running the installation interactively to rule out the possibility that the problems you are encountering are not related to silent mode. |

Creating response files for iSeries Access for Windows installations

A response file records the selections made in response to the prompts in the installation process. During a silent installation, the setup program will use the response file to get the information necessary to complete the installation.

To create a response file, follow these steps:

1. At the command line in the iSeries Access for Windows installation image directory on a PC, type:

```
setup -r -f1d:\dir\file.iss
to run an installation, and record the responses.
```

-

- **-f1** is an optional parameter used to indicate an alternate response file name. If you do not use this parameter, then setup.iss records all of the installation choices. Setup.iss resides in the Windows directory, for example, C:\Windows or C:

- **d:\dir** is the drive and directory where you want to create the response file. If you use the **-f1** parameter, then you must specify the drive and directory along with the response file name that you want to create.
- **file.iss** is the name of the response file that you want to create. The file extension must always be iss.

2. Complete the setup program, providing the responses you want to use during the silent installations.

After the installation is complete, the iss file that is created will look somewhat like this example response file.

Starting a silent installation

Silent installations use a response file (file.iss) for the responses to prompts during the installation process. This eliminates the need for any user interaction during the installation process, and allows you to quickly and easily copy duplicate installations across your network. Information about the status of the silent installation can be recorded in a log file (file.log).

To start a silent installation, type the following at a command prompt in the iSeries Access for Windows installation image directory:

```
setup -s -f1d:\dir\file.iss -f2d:\dir\file.log
```

where:

- **-f1** is an optional parameter where you can specify the response file (**file.iss**) to use. If you do not use this parameter, then the installation attempts to use a default response file named setup.iss. It looks for

this file in the directory containing setup.exe. **d:\dir** is the drive and directory that contains the response file that you want to use. If you use the **-f1** parameter, then you must specify the drive and directory along with the response file name.

- **-f2** is an optional parameter where you can specify the location and name for the log file that the silent installation creates. If you do not use this parameter, the installation creates a log file named setup.log and places it in the directory containing setup.exe. **d:\dir** is the drive and directory that contains the log file. If you use the **-f2** parameter, then you must specify the drive and directory along with the log file name. **file.log** is the name of the log file that you want to create.

Return codes for silent installations or migrations

To see if your silent installation was successful, look at the return codes in the log file. If you receive a return code of 0, the installation was successful. If the return code was not 0, take any action necessary to resolve the problem. You specified the name and location of the log file when you started the silent installation. You can also see additional information on failures in silent.txt in the target directory, or in cwbsilent.txt in the windows directory (Windows or Winnt) if the target directory is not yet set.

| return code | meaning |
|-------------|---|
| 0 | Success |
| -1 | General error |
| -2 | Invalid mode |
| -3 | Required data not found in the Setup.iss file |
| -4 | Not enough memory available |
| -5 | File does not exist |
| -6 | Cannot write to response file |
| -7 | Unable to write to the log file |
| -8 | Path to the InstallShield silent response file is not valid |
| -9 | Not a valid list type (string or number) |
| -10 | Data type is not valid |
| -11 | Unknown error during set up |
| -12 | Dialog boxes are out of order |
| -51 | Cannot create the specified folder |
| -52 | Cannot access the specified file or folder |
| -53 | A selected option is not valid |

For more information, refer to Starting a silent installation.

Administering service packs

Fixes for iSeries Access for Windows are integrated into service packs, which are packaged into a Program Temporary Fix (PTF) for delivery. You can download the latest PTF to your iSeries server to provide a more stable operating environment for the iSeries Access for Windows client, and to correct known problems. Once you have installed the PTF on your host system, you can use Check service level to distribute service packs to client PCs.


Obtain the latest PTF for installation on your iSeries server

Use SNDPTFORD to order the PTF for your iSeries server. Since the service pack PTFs generally exceed the size limit to be sent electronically, you can receive the PTF on media by changing the Delivery Method, DELIVERY, parameter on SNDPTFORD to *ANY. (The parameter defaults to *LINKONLY.) Alternatively, use Internet PTF Delivery (iPTF). To find out about this service and the requirements, go to

iSeries Technical Support  , and select **Fixes and Updates** from the left menu.

Install service packs directly on the client PCs

You can also download service packs to your client PCs. This allows you to update certain client PCs without applying the PTF to your host. To obtain the latest service pack, see the iSeries Access home

page  and select **Service Pack** from the table of links. After downloading the service pack, simply run the setup file to perform the upgrade. You should always reboot after installing a service pack.

Service pack PTFs update the iSeries Access for Windows installation image on the iSeries server. All installations will reflect the host iSeries server's latest service pack level.

Note: On Windows NT/2000/XP, only users with administrator security can perform service pack and iSeries Access for Windows upgrades. You can circumvent Windows NT/2000/XP administrator security to let users apply service packs without administrator authority.

Servicing other components and third party applications

Check Service Level also manages the version of other components, like SSL, and third party applications (plug-ins and add-ins). Check Service Level automatically checks the host iSeries server for updates to any installed components. If updates are available, the user will usually be alerted, and asked to allow the update. This opens Selective Setup in a special mode and updates the appropriate component.

Check service level

You can use iSeries Access for Windows Check Service Level on the PC to detect updates to iSeries Access for Windows and related components on the iSeries server. To define options for running check service level, go to the **Service** tab of **iSeries Access for Windows Properties**.

From there you can set the following parameters:

- When to have check service level run
- A date to check service level
- The number of days before checking service level
- The number of minutes to delay (after log-on) to check service level

Note: Policies may dictate what you can do with the above functions. For example, you can force the number of days before checking service level to be a certain value. Done this way, users could not alter this value. You can also use Application Administration to dictate options with the above functions.

You can also select to

Installing the service pack silently

Check the **Perform silent installation** box on the **Service** tab of **iSeries Access for Windows Properties** to do service level checks and service pack installation silently, without any user interaction. The silent service pack installation utility will use information from a response file to answer prompts automatically.

The response file is identical to the one used in silent installation, except you must specify the following name.

- SLTSP.ISS - for service packs (This file must reside in the same directory as your service pack **setup.exe** does)
- SLTUP.ISS - for upgrades (This file must reside in the same directory as your installation **setup.exe** does)

When you create your response file, you can set a parameter to reboot automatically. If you set this to yes, you should set SCHEDCHECK in a scheduled job so that the silent check service version runs during the night. See the online iSeries Access for Windows User's Guide for more information about SCHEDCHECK.

If set to no, a message box appears asking the user to **OK**

ODBC administration

Open Database Connectivity (ODBC) is a Microsoft standard for providing access to databases. It has a well-defined set of application programming interfaces (APIs) that use Structured Query Language (SQL) to access databases.

iSeries Access ODBC driver overview

This topic provides a general description of ODBC, and how you can use it with iSeries Access for Windows.

Setting up your system for the ODBC driver

This topic presents procedures for setting up your environment to support the ODBC driver. For help configuring the ODBC driver, start the ODBC administration program from the iSeries Access for Windows program group, and refer to the online help.

Security considerations for ODBC

This topic highlights a few security considerations when working with ODBC, and provides references to more detailed security instructions.

Troubleshooting ODBC

This topic can help you solve a few of the more commonly encountered difficulties with iSeries Access for Windows and ODBC. It also identifies several tools that can help you remove performance bottle necks. You should review this information before contacting technical support.

iSeries ODBC Driver for Linux

This topic discusses installing Linux on an iSeries logical partition and using the iSeries ODBC Driver for Linux to access the iSeries database.

Note: iSeries ODBC Driver for Linux is not part of iSeries Access for Windows. It is a separate product used only with the Linux operating system.

For help with integrating ODBC support into your applications, refer to the iSeries Access for Windows ODBC programming, where you can get information on the following sub topics:

- ODBC API list
- ODBC API implementation issues
- Programming examples
- ODBC performance

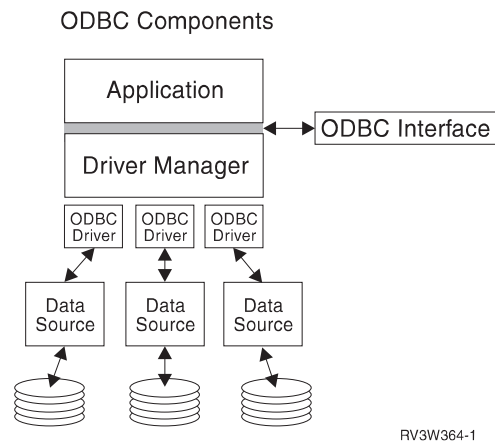
Overview of the iSeries Access ODBC driver

The iSeries Access ODBC driver is a collection of application programming interfaces (APIs) for accessing database information using Structured Query Language (SQL). Using the iSeries Access ODBC driver allows applications to access different databases on the iSeries server using the same source code, and to handle data in the format most convenient for those applications. ODBC provides an application developer a relatively simple model for creating portable applications or components that must deal with multiple DBMSs.

The ODBC architecture involves an application, driver manager, ODBC driver, and a data source. iSeries Access provides both a 32-bit and 64-bit ODBC driver. The 64-bit ODBC driver is automatically installed along with the 32-bit ODBC driver when running under a 64-bit version of Windows. ODBC applications

running in 64-bit versions of Windows will automatically use the appropriate ODBC driver, depending on what bit version the application was compiled for. For example, the 64-bit driver can only be used by a 64-bit application.

In order for an application to use ODBC you must set up a data source. You can use the ODBC Administrator to set up a data source. There are two versions of the ODBC Administrator, 32-bit and 64-bit, that can be accessed from the iSeries Access for Windows folder. When using ODBC Administrator, you have the option to setup three different types of data sources: User, System, and File data sources. For more information about how data sources are configured, see 64-bit ODBC Support, in the iSeries Access for Windows' User's Guide.



Application Performs processing and calls ODBC functions to run SQL statements.

Driver manager Processes ODBC function calls and forwards the requests to the driver.

Driver Processes ODBC function calls, submits SQL requests to a specific data source, and returns results to the application.

Data source To use a data source you have to create a Data Source Name (DSN). A DSN contains information about how to access the DBMS. You can specify any of the following DSNs:

- *User DSN*: These data sources are local to a computer, and may only be available to the user who created them. This information is stored in the registry.
- *System DSN*: These data sources are local to a computer, rather than dedicated to a user. The system, or any user having privileges, can use a data source set up with a system DSN. This information is stored in the registry.

Note: On a PC with a 64-bit processor, the system part of the registry is split into 32-bit and 64-bit pieces. System DSNs configured using the 32-bit ODBC Administrator are available only to 32-bit applications. Also, System DSNs configured using the 64-bit ODBC Administrator are available only to 64-bit applications.

- *File DSN*: These are file-based data sources that may be shared between all users that have the same drivers installed so that they have access to the database. These data sources do not need to be dedicated to a user, or to be local to a computer.

For more information about ODBC, refer to the Microsoft web site.

Setting up your system for the iSeries Access ODBC driver

The iSeries Access ODBC driver is an ODBC version 3.5 compliant driver. The driver requires Microsoft Data Access Components (MDAC) version 1.5 or higher. Applications that use Microsoft ActiveX Data Objects (ADO) should have MDAC version 2.1 or higher installed. The runtimes for MDAC versions 2.1

and later provide additional function for applications that use ADO, the Microsoft OLEDB provider for ODBC, and iSeries Access for Windows ODBC, to access their iSeries data. If an application uses connection pooling or Microsoft Transaction Server (MTS) support, we recommend that the latest MDAC version be installed. You can download MDAC from the following Microsoft Web Site:

<http://www.microsoft.com/data>  .

Before configuring the ODBC driver, you need to set up your system.

See the following to set up your system for the iSeries Access ODBC driver:

1. Add the local system to the relational database (RDB) directory on OS/400.
2. Set up an ODBC data source.

The following is optional and may require additional set up steps:

Independent ASPs

To use Independent ASPs through ODBC, configure your ODBC DSN and do the following:

1. Select the **Server** tab.
2. Specify the **RDB name** that corresponds with the Independent ASP that you wish to connect to.
3. If no RDB name is specified, the default RDB name is determined from the job description of the user profile that is making the ODBC connection. By default, the driver uses the setting of the user profile for the user making the ODBC connection.

For more information about Independent ASPs, see Independent ASPs.

For help configuring options for a specific data source, start the ODBC Administrator from the iSeries Access for Windows program group, select the data source to configure, and refer to the online help.

Adding the local system to the RDB directory

To use ODBC, the local system name must appear in the RDB directory.

To add the local system to the RDB directory:

1. From the command prompt run the CL command, Add Relational Database Directory Entry (ADDRDBDIRE).
2. When the ADDRDBDIRE screen prompts you for values, enter the name of the system as the Relational Database parameter.
3. Enter *LOCAL as the Remote Location parameter.

There may be additional steps to get the database (RDB) name set, if the version of your system is V5R2 or later and your application accesses data in Independent ASPs. The RDB name corresponds with a namespace that consists of the system ASP and any user ASPs or linked ASP group associated with the system ASP. For more information about Independent ASPs, see Independent ASPs.

Note:

ODBC allows the use of fully qualified names in the format of [catalog name].[schema name].identifier (for example, where identifier is the name of a table, view, or procedure). In the DB2/400 implementation of SQL this corresponds to [RDB bane].[collection name].identifier.

Specify the ODBC data source

You must specify the data source for your application to access and manipulate data.

To specify the data source:

1. Start the ODBC Administration program from the iSeries Access for Windows program group.
2. Select the appropriate tab for the type of data source. See ODBC overview for more information.
3. Select an existing data source from the list, or select **Add** to create new one. If you are using an existing data source, click **Configure** and proceed to step 5.
4. Select the iSeries Access ODBC driver for your data source, and click **Finish**.
Note: You may notice 'Client Access ODBC Driver (32-bit)' in the list of drivers. This is here so that data sources created with previous releases of Client Access will work. Both names point you to the same ODBC driver. You can use either name, however in future releases the 'Client Access ODBC Driver (32-bit)' will be removed.
5. Specify desired options using the iSeries Access for Windows ODBC setup dialog. For a description of the controls, refer to the data source's online help by using the F1 key or the Help button.

Note:

The data source name can include up to 32 characters, must start with an alphabetic character, and cannot include the following characters:

Unallowed data-source characters

Left bracket ([)

Question mark (?)

Right bracket (])

Asterisk (*)

Left brace ({)

Equal sign (=)

Right brace (})

Exclamation point (!)

Left parenthesis (()

At sign (@)

Right parenthesis ())

Semicolon (;)

iSeries Access for Windows ODBC security

The following information is not intended to be a comprehensive guide to security strategies on the iSeries servers or with iSeries Access for Windows. It simply provides an overview of security strategies that impact iSeries Access for Windows and ODBC users. For more in-depth information, see the IBM Security

- Reference  .

- Risky ODBC security strategies
- ODBC program security strategies
- Other information resources for ODBC security

Risky ODBC security strategies

Some system administrators attempt to secure access to the data, rather than securing the data itself. This is extremely risky, as it requires that administrators understand ALL of the methods by which users can access data. Some common ODBC security techniques to avoid are:

Command Line Security

This may be useful for "green-screen" or 5250 emulation-based applications. However, this method assumes that if you prevent users from entering commands in a 5250 emulation session, they can access data only through the programs and menus that the system administrator provides to them. Therefore, command line security is never truly secure. The use of iSeries Access policies and application administration improves security, and use of object level authority improves it even more.

Potentially, iSeries Access for Windows Policies can restrict ODBC access to a particular data source that might be read only. Application Administration in iSeries Navigator can prevent ODBC access.

For additional information, see the IBM Security - Reference .

User exit programs

A user exit program allows the system administrator to secure an IBM-supplied host server program. The iSeries Access ODBC driver uses the Database host server: exit points QIBM_QZDA_INIT; QIBM_QZDA_NDBx; and QIBM_QZDA_SQLx. Some ODBC drivers and iSeries Access for Windows data access methods (such as OLE DB) may use other host servers.

Journals

Journaling often is used with client/server applications to provide commitment control. The journals contain detailed information on every update made to a file that is being journaled. The journal information can be formatted and queried to return specific information, including:

- The user profiles that updated the file
- The records that were updated
- The type of update

Journaling also allows user-defined journal entries. When used with a user exit program or trigger, this offers a relatively low-overhead method of maintaining user-defined audits. For further information, see the

Backup and Recovery .

Data Source Name (DSN) restrictions

The iSeries Access ODBC driver supports a DSN setting to give read-only access to the database. The iSeries Access ODBC driver supports a read-only and a read-call data source setting. Although not secure, these settings can assist in preventing inadvertent delete and update operations.

ODBC program security strategies

Consider the following ODBC program security strategies.

Restricting program access to the database

System administrators often need to limit access to particular files, to a certain program, or to sets of programs. A "green-screen" programmer would carry out restrictions by using program-adopted authority. A similar method can be used with ODBC.

Stored procedures allow ODBC programmers to implement program-adopted authority. The programmer may not want users to be able to manipulate database files by using desktop applications such as Microsoft Access or Lotus 1-2-3. Instead, the programmer may want to limit database updates to only the programmer's application. To implement this, user access to the database must be restricted with object-level security or with user exit programs. The application must be written to send data requests to the stored procedure and have the stored procedure update the database.

Restricting CPU utilization by user

ODBC has greatly eased the accessibility of iSeries data. One negative impact has been that users may accidentally create very CPU-intensive queries without realizing it. ODBC runs at an interactive job priority and this can severely affect system performance. The iSeries supports a **query governor**. ODBC can invoke the query governor (for example, through the PC application) in a stored procedure call. Or the ODBC APIs can invoke the governor by way of the query time-out parameter. Also, a user exit program can force the query governor on the ODBC job. The time limit is specified on the QRYTIMLMT parameter of the CHGQRYA CL command. The query options file (QAQQINI) can also be used to set the value.

The *SQL Reference* book contains additional information. View an HTML online version of the book, or print a PDF version, from the DB2 Universal Database for iSeries books online.

Also see Administering Client Access Express host servers, for more information.

Audit logs (monitoring security)



Several logs can be used to monitor security. QHST, the History Log, contains messages that relate to security changes that are made to the system. For detailed monitoring of security-related functions, QAUDJRN can be enabled. The *SECURITY value logs the following functions:

- Changes to object authority
- Create, change, delete, display, and restore operations of user profiles
- Changes to object ownership
- Changes to programs (CHGPGM) that adopt the owner's profile
- Changes to system values and network attributes
- Changes to subsystem routing
- When the QSECOFR password is reset to the shipped value by DST
- When the DST security officer password is requested to be defaulted
- Changes to the auditing attribute of an object

For additional information, see the IBM Security - Reference .

Related information for ODBC security

In-depth security reviews and assistance to implement the strategies above is available through IBM Consultline (1-800-274-0015). Please review the following for in-depth information on specific topics:

- Administering host servers
- IBM Security - Reference 
- Backup and Recovery 
- DB2 Universal Database for iSeries

Troubleshoot ODBC

The following topics provide general guidelines for finding and resolving iSeries Access for Windows ODBC errors:

- ODBC diagnostic and performance tools
- Error messages
- Troubleshoot the iSeries server connection
- Common ODBC errors
- Gathering information for IBM Support

ODBC diagnostic and performance tools

The following tables contain ODBC diagnostic and performance tools for both the client and server sides:

Client-side tools

| | |
|----------------------|--|
| ODBC Trace (SQL.LOG) | Microsoft's ODBC Administrator provides its own trace utility to trace ODBC API calls from applications. |
| ODBC trace utilities | See Collecting an ODBC Trace (SQL.LOG), for more information. There are other ODBC trace utilities available that can be more robust than the ODBC Trace (SQL.LOG). These retail utilities can provide detailed entry and exit point tracing of ODBC API calls. Two tracing utilities are: Trace Tools (Dr. DeeBee) and SST Trace Plus (Systems Software Technology). |
| CWBPING | To use CWBPING, type <code>cwbping (your system name or IP address)</code> at an MS-DOS prompt. For example: <code>cwbping testsys1</code> or <code>cwbping 127.127.127.1</code> CWBPING responds with a list of servers, and their status. Run CWBPING without any parameters for help with using CWBPING. For more information about CWBPING, see checking the server status. |
| CWBCOTRC | To use CWBCOTRC, type CWBCOTRC ON at an MS-DOS prompt while located in the <code>\Program Files\IBM\Client Access</code> directory. After turning on the trace, you can start your application. Typing CWBCOTRC OFF stops tracing. CWBCOTRC gathers information about data that is being transmitted to and from the server. Run CWBCOTRC without any parameters for help with using CWBCOTRC. |
| Detail trace | Detail trace gathers information traced out by the iSeries Access for Windows components that are in use. ODBC information that can be found in this trace includes entry points into the driver, information about the prestart job, the package name in use, and special error conditions. For more information, see Gathering a detail trace. |

Server-side tools

Communications trace The communications trace facility will trace and format any communications type that has a line description (token ring and Ethernet).

This is a tool for isolating many problems. It also is a useful aid for diagnosing where a performance delay is occurring. Use the timestamp and eye-catcher fields to measure how long it takes to process a request.

Job traces The job trace can help isolate most host problems and many performance issues. A service job must first be started on the job to be traced. Locate the fully qualified job name of the ODBC job. From any 5250 emulation session, start a service job on this QZDASOINIT job by using the STRSRVJOB command. Then choose one of two traces, depending on the information needed:

Trace job

Traces the internal calls made by the host server. Run the `TRCJOB *ON` command.

Debug trace

Used to review the performance of your application and to determine the cause of a particular problem.

The STRDBG command can be run against an active service job. This command logs the decisions made by the query Optimizer to the job log of the debug session. It records estimated query times, access paths used, cursor errors, etc. Activate STRDBG from the **Diagnostic** tab of the DSN setup dialog in **ODBC Administration** or use the following command:

```
STRDBG UPDPROD(*YES)
```

The ODBC job log can record all errors that occur on the iSeries server. When the job is in debug mode, the job log also will contain performance-related information.

Performance toolkit provides reports and utilities that can be used to create an in—depth analysis of your application performance. The toolkit provides information about CPU utilization, disk arm utilization, memory paging and much more. Although the base operating system includes the ability to collect performance data, you will need the separately licensed program **Performance Tools/400** to analyze the results.

You can also use the tools Database Monitor and Visual Explain. Refer to the iSeries Navigator Online help for more information.

QZDASOINIT Receive optimal support, generate, locate and retrieve the QZDASOINIT job log. The job log may contain job messages that can help you to determine and resolve errors that are returned through ODBC.

log You can generate and find a job log from the datasource. Use the diagnostics tab option **Print job log at disconnect** to generate a job log. To find the job log, open a PC5250 emulation session and issue a WRKSPLF where user is the iSeries user profile used on the ODBC connection.

QAQQINI The query options file contains many options that can be used for diagnostics, tuning, and debug. Refer to the (Query database documentation for details. You can also set this file in the ODBC data source (DSN).
options
file)

iSeries Access ODBC error messages

When an error occurs, the iSeries Access ODBC driver returns the SQLSTATE (an ODBC error code) and an error message. The driver obtains this information both from errors that are detected by the driver and from errors that are returned by the DBMS.

For errors that occur in the data source, the iSeries Access ODBC Driver maps the returned native error to the appropriate SQLSTATE. When both the iSeries Access ODBC driver and the Microsoft Driver Manager detect an error, they generate the appropriate SQLSTATE. The iSeries Access ODBC driver returns an error message based on the message returned by the DBMS.

For errors that occur in the iSeries Access ODBC driver or the Microsoft Driver Manager, the iSeries Access ODBC driver returns an error message based on the text associated with the SQLSTATE.

Error message format

Error messages have the following format:

```
[vendor] [ODBC-component] [data-source]
error-message
```

The prefixes in brackets ([]) identify the source of the error. The following table shows the values of these prefixes returned by the iSeries Access ODBC driver.

When the error occurs in the data source, the [vendor] and [ODBC-component] prefixes identify the vendor and name of the ODBC component that received the error from the data source.

| Error Source | Value |
|----------------------------|--|
| Driver Manager | [Microsoft] [ODBC driver Manager] [N/A] |
| iSeries Access ODBC driver | [IBM] [iSeries Access ODBC driver] N/A |
| NLS messages | [IBM] [iSeries Access ODBC driver] Column #: NLS error message number NLS error message text |

| Error Source | Value |
|---------------------|--|
| Communication layer | [IBM] [iSeries Access ODBC driver] Communications link failure.Comm RC=xxxx - (message text) Where xxxx is the error number in decimal, not hexadecimal, format. Message text describing the nature of your error appears with the error number. Note: For more information about error message ids,see iSeries Access return codes or the iSeries |
| DB2 UDB for iSeries | [IBM] [iSeries Access ODBC driver] [DB2 UDB] Server error message |

Viewing DB2 UDB for iSeries error message text:

| For errors that begin with: | Use this OS/400 command |
|-----------------------------|---|
| SQL | DSPMSGD RANGE(SQLxxxx) MSGF(QSQLMSG) |
| IWS or PWS | DSPMSGD RANGE(ZZZxxxx) MSGF(QIWS/QIWSMSG) where ZZZ is IWS or PWS |

Refer to Common ODBC errors for help with other ODBC error messages.

You can search and view NLS or communication error messages in the iSeries Access for Windows online User's Guide, in the Service, Error and Trace message help topic.

Troubleshooting the iSeries server connection

Each ODBC connection communicates with one database server program that runs on the iSeries server. This program is referred to as the **host server program**. The name of the Database Server program used with TCP/IP is **QZDASOINIT**. It is normally located in subsystem QSYS, however it can be set up differently by the system administrator.

Under normal conditions, the program is evoked transparently, and the user is not required to take action except to verify that the proper subsystems and communication protocols are running. See the iSeries Access for Windows Host Server administration for details on administration of host server jobs.

The most common indication of a connection failure is an error message from the ODBC driver mentioning a communications link failure.

If ODBC is unable to connect to the iSeries server, perform the following troubleshooting tasks:

- Check the server status
- Verify that the proper subsystems are running
- Verify that the proper prestart jobs are running
- Additional TCP/IP considerations

Checking the server status: The iSeries Access for Windows product has a special command to verify status of host servers:

```
CWBPING systemname
```

where systemname is the name of the system.

The command should return something like the following:

```

To cancel the CWBPING request, press CTRL-C or CTRL=BREAK
I - Verifying connection to system MYSYSTEM...
I - Successfully connected to server application: Central Client
I - Successfully connected to server application: Network File
I - Successfully connected to server application: Network Print
I - Successfully connected to server application: Data Access
I - Successfully connected to server application: Data Queues
I - Successfully connected to server application: Remote Command
I - Successfully connected to server application: Security
I - Successfully connected to server application: DDM
I - Successfully connected to server application: Telnet
I - Successfully connected to server application: Management Central
I - Connection verified to system MYSYSTEM

```

Notes:

- For ODBC to work, the database and security servers must be operational.
- If a message is displayed indicating that the connection is configured to use SSL, the connection may only be used by 32-bit applications. Use of the connection through the 64-bit iSeries Access ODBC driver or the 64-bit iSeries Access OLE DB provider will fail. To successfully connect to an iSeries server using a 64-bit application, you must first configure that connection to not use SSL.

Verifying that subsystems are active: TCP/IP-connected ODBC jobs (QZDASOINIT) will run in the QSERVER subsystem. Verify that this subsystem is running. The QSERVER subsystem may need to be manually started. To do this, simply issue the following command:

```
STRSBS QSERVER
```

To have the subsystem start automatically at IPL, then modify the IPL Start up procedure (the default is QSYS/QSTRUP) to include the STRSBS QSERVER command.

In addition to subsystem QSERVER, subsystem QSYSWRK must be running.

Verifying that prestart jobs are running: IBM ships the QSERVER subsystem configured to use prestart jobs to improve performance at job initialization/start up. When prestart jobs are configured in the subsystem, the job **MUST** be active to connect. The prestart job used for a TCP/IP connection is:

- QZDASOINIT - Server program

To verify a prestart job is running:

```
WRKACTJOB SBS(QSERVER)
```

The appropriate prestart jobs should be active:

| Job | User | Type | -----Status----- | |
|------------|-------|------|------------------|---------------------|
| QZDASOINIT | QUSER | PJ | ACTIVE | (socket connection) |
| QZDASRVSD | QUSER | PJ | ACTIVE | (socket connection) |

Prestart jobs do not display in WRKACTJOB unless a connection is already active. You must use F14 - Include from the WRKACTJOB panel

Additional TCP/IP considerations: Verify that TCP/IP is started with the following command:

```
NETSTAT *CNN
```

Note: To verify that TCP/IP is started with iSeries Navigator, you must already have configured your server with TCP/IP, then do the following:

1. In iSeries Navigator, select your server —> Network.
2. Right-click TCP/IP Configuration, and select Utilities.
3. Select Ping.
4. Specify a host name or TCP/IP address, and click Ping Now.

Use the command STRTCP to start the desired protocol if it is not running.

Verify the necessary daemons are running by browsing the information returned from the NETSTAT *CNN command:

| Remote Address | Remote Port | Local Port | Idle Time | State |
|----------------|-------------|------------|-----------|--------|
| * | * | as-cent > | 000:09:31 | Listen |
| * | * | as-signon | 000:09:41 | Listen |
| * | * | as-svrmap | 002:57:45 | Listen |
| * | * | as-data > | 002:57:45 | Listen |

Use the command STRHOSTSVR SERVER(*ALL) to start them if necessary.

- Verify QZDASRVSD, the ODBC socket daemon, is running.
 - as-database should be in the Listen State
 - WRKJOB QZDASRVSD should be used to check the job log of the daemon for any error messages.
- Verify that socket daemon QZSOMAPD is running in QSYSWRK subsystem.
 - as-svrmap should be in the Listen State as shown by NETSTAT *CNN.
 - WRKJOB QZSOMAPD should be used to check the job log of the daemon for any error messages.

The PC locates the socket used by the database server by connecting to the server mapper socket. It retrieves the socket used by as-database. It then connects to the proper socket which is being monitored by the file server daemon, QZDASRVSD. The server daemon will attach the client's connection to a QZDASOINIT prestart job in QSERVER. After validating the user profile and password and swapping the user profile into the prestart job, the job will run similar to the QZDASOINIT job. If this is the first connection made to the for this PC, then two other servers are used: Central server for licensing and signon server for userid/password validation.

For more information about verifying that TCP/IP is started, see General TCP/IP problems.

Common ODBC errors

The following topics provide general guidelines for finding and resolving common iSeries Access for Windows ODBC errors:

- SQL errors
- Stored procedure errors
- ODBC incorrect output and unpredictable errors

SQL errors:

- SQL0113 - Name &1 not allowed.
- SQL0114 - Relational database &1 not the same as current &2 server
- SQL0204 - MYSYSCONF not found
- SQL0208 - ORDER BY column not in result table
- SQL0900 - Application process not in a connected state
- SQL0901 - SQL System Error

- SQL5001 - Column qualifier or table &2 undefined.
- SQL5016 - Object name &1 not valid for naming convention
- SQL0104 - Token &1 was not valid. Valid tokens: &2
- SQL7008 &1 in &2 not valid for operation. The reason code is 3

Note: For more information on SQL errors, see DB2 Universal Database for iSeries SQL Messages and Codes.

Stored procedure errors: The following are typical stored procedure errors:

- SQL0444 - External program &A in &B not found (DB2 UDB for iSeries SQL)
- No data returned on OUTPUT and INPUT_OUTPUT parameters
- SQL0501 - Cursor CRSR000x not open

SQL0444 - External program &A in &B not found (DB2 UDB for iSeries SQL): The SQL0444 is generated on an execute or execute direct when the database server is able to locate the procedure declaration but is unable to locate the program object. The external program must be in the location specified in the system catalog tables. Note that this location is defined by the naming convention and default collection in affect when the procedure is defined (using CREATE PROCEDURE) and not when the procedure is called. To check the location defined for the external program name of a stored procedure run a query over QSYS2.SYSPROCS and note the value for the "EXTERNAL_NAME" name field.

No data returned on OUTPUT and INPUT_OUTPUT parameters: This problem could be caused by any of the following:

- The ODBC **SQLBindParameter** API incorrectly specified **fParamType** as SQL_PARAM_INPUT.
- DECLARE PROCEDURE was used instead of CREATE PROCEDURE, and extended dynamic support is disabled.
- The programmer incorrectly declared a parameter as IN on the CREATE or DECLARE PROCEDURE.
- The stored procedure program incorrectly returned the parameter.
-

SQL0501 - Cursor CRSR000x not open: To return data when using embedded SQL in ILE programs, you must specify the compile option ACTGRP(*CALLER) and not the default of *NEW.

Verify that the program executes a return instead of an exit.

When the stored procedure program executes an exit instead of a return, you must set the **Close SQL Cursor** option to *ENDACTGRP. If the Close SQL Cursor option is set to *ENDMOD, the cursor will be closed before data is retrieved.

Also verify that the CREATE PROCEDURE specifies the correct number of result sets. This is especially important when using array result sets.

ODBC incorrect output and unpredictable errors: Ensure that the iSeries Access ODBC driver and the database server program are at matching code levels. Check for PTF co-requisite requirements on any PTF that you order or in the readme.txt file of the Service Pack. If problems continue, verify that you have disabled the prefetch option in the ODBC Data Source. The prefetch option should not be used if the application uses either the SQLExtendedFetch or SQLFetchScroll ODBC API, or if you are not sure.

Note that stored procedure result set cursors are forward only, read only.

Binary or hexadecimal data instead of ASCII characters

The default value of the Translation parameter is set to not convert binary data (CCSID 65535) to text. The CCSID is attached to files, tables and even fields (columns). This CCSID determines

which conversion table will be used to convert data, for example from EBCDIC to ASCII. A CCSID of 65535 often identifies raw data (binary or hexadecimal), such as bitmapped graphics, that is language independent. Not selecting Convert binary data (CCSID 65535) to text ensures that the raw data is not damaged.

Setting the translation parameter to Convert binary data (CCSID 65535) to text updates the CCSID that is attached to the data to the CCSID of the job. This parameter setting can cause damage to the data, if the data is truly binary.

Gathering information for IBM Support

So the IBM Support staff can offer you the best service, please have certain information available when you open a problem record to IBM Support. To gather this information, complete the following tasks:

| | |
|---|--|
| Record the OS/400 version and cumulative PTF level. | <ol style="list-style-type: none"> 1. Issue the display PTF command on an terminal emulation command line: DSPPTF 2. Record the OS/400 release information that has the format VxRxMx. 3. Verify that the IPL source is ##MACH#B. 4. Press F5 to display the PTF details. 5. Record the first PTF ID in the list. It will have the format Tzxyyy where xx is the year, yyy the Julian date and z is either L or C. |
| Record the version of the ODBC driver. | <ol style="list-style-type: none"> 1. From the Task bar select Start -> Programs -> IBM iSeries Access for Windows -> ODBC Administration. Note: On a 64-bit machine using a 64-bit driver, select ODBC Administration (64-bit). 2. Select the Drivers tab. 3. Record the version of the iSeries Access ODBC Driver. |
| Record the version of the ODBC driver manager. | <ol style="list-style-type: none"> 1. From the Task bar select Start -> Programs -> IBM iSeries Access for Windows -> ODBC Administration. Note: On a 64-bit machine using a 64-bit driver, select ODBC Administration (64-bit). 2. Select the About tab. 3. Record the version of the Driver Manager. |
| Gather traces | The traces you will most likely be asked to gather for support are: an ODBC trace (SQL.LOG), CWBCOTRC or Communication Trace, and a Detail Trace. See ODBC diagnostic and performance tools, for more information about traces. |
| Record additional information | Such as the PC application, the error description, and what ODBC driver (32-bit or 64-bit) you are using. |

Host server administration

This topic provides brief descriptions of server functions that run on an iSeries server and technical information specific to host servers that are used by the iSeries Access for Windows product. These are not all of the servers used by iSeries Access for Windows, and this topic does not address all of the servers on the host (iSeries) system.

OS/400 host servers

Host servers handle requests from client PCs or devices such as running an application, querying a database, printing a document, or even performing a backup or recovery procedure. iSeries computers are full-function servers capable of performing many tasks at once, including file, database, applications, multimedia, mail, print, fax, and wireless communications. When these tasks are handled by several different servers, server management and coordination becomes complex. Having all of your servers on one integrated system greatly reduces the overall cost and complexity of managing your network.

These servers are used by iSeries Access for Windows, but are designed so that other client products can also use them. This topic focuses on how these servers are used by iSeries Access for Windows.

Adding or removing the OS/400 Host Server option

The OS/400 servers discussed here are all optimized servers, and are included with the base option of OS/400. To use the iSeries Navigator function of iSeries Access for Windows, install the Host Server option.

If you are not using any iSeries Access for Windows products or iSeries NetServer and would like to remove the OS/400 Host Server option, you should end the subsystems used by these servers before you remove the option. End the QBASE or QCMN subsystem (for host servers with APPC support), the QSYSWRK and QUSRWRK subsystems (for host servers with sockets support), and the QSERVER subsystem (for database and file server). Problems may occur if you try to delete the option while any of these subsystems are active.

- **OS/400 host servers**
This topic describes many of the host servers that are common in the iSeries Access for Windows client and the related objects. You can view the servers by type or by their function in iSeries Access for Windows.
- **Using host servers**
This topic describes the client/server communication process, and how to manage it. Additionally, this topic lists relevant iSeries system values and subsystems, and described how to identify, display and manage server jobs on the iSeries.
- **Using exit programs**
This topic shows how to write and register exit programs. You can also find exit program parameters and programming examples in this section.

OS/400 host servers

This information covers only the servers used by iSeries Access for Windows. This does not include all of the servers on the host (iSeries) system. iSeries Access for Windows host servers include:

Host servers by iSeries Access for Windows function

Host servers listed by their associated function in iSeries Access for Windows.

File server

The file server allows clients to store and access information, such as files and programs, located on the iSeries server.

Database server

For Data Transfer, ODBC, iSeries Navigator database, SQL APIs (DB APIs) and the iSeries Access for Windows OLE DB provider.

Data queue server

Provides access to data queues on the iSeries server.

Network print server

Provides remote print support and additional print management functions.

Central server

Provides services such as license management and other client management functions.

Remote command and program call server

Allows PC applications to issue commands and call programs in OS/400 and return the results to the client.

Signon server

Provides password management functions for host servers with sockets support.

Server port mapper

Provides the current server port number to a client requesting a connection.

Host servers by iSeries Access for Windows function

The following table shows a subset of the servers that are used with some of the functions in iSeries Access for Windows.

| Client Function | OS/400 Server Used |
|---|---|
| Database access APIs | Database Server |
| <ul style="list-style-type: none"> • SQL • ODBC APIs | |
| Data Transfer | Database Server |
| ODBC driver | Database Server |
| Access Integrated File System from iSeries Navigator | File Server |
| Data queue APIs | Data Queue Server |
| OLE DB provider | <ul style="list-style-type: none"> • Data Queue Server • Database Server • Remote Command and Distributed Program Call Server • Signon Server |
| License management | Central Server |
| <p>Done when an application that requires a license is started (Data Transfer and 5250 emulation)</p> | |
| Retrieve conversion map | Central Server |
| <p>Done only on initial connection if the client does not contain the required conversion maps</p> | |
| Remote command functions | Remote Command and Distributed Program Call Server |
| Distributed program call | Remote Command and Distributed Program Call Server |
| Send password for validation and change expired password (TCP/IP) | Signon Server |
| Network Print | Network Print Server |
| GUI and programming interfaces. | |

For more information, refer to iSeries Access for Windows Servers and Ports Required, APAR

File Server

The file server allows clients to store and access information, such as files and programs, located on the iSeries server. The OS/400 file server interfaces with the integrated file system on the iSeries server. Clients use their own interface to interact with the file systems, rather than the integrated file system user interfaces and APIs.

The integrated file system is a part of the OS/400 program. It supports stream input/output and storage management similar to personal computer and UNIX operating systems. At the same time, it integrates all information that is stored on the iSeries server.

The key features of the integrated file system are the following:

- Support for storing information in stream files, which are files that contain long, continuous strings of data. These strings of data might be, for example, the text of a document or the picture elements in a picture. Documents that are stored in iSeries folders are stream files. Other examples of stream files are PC files and the files in UNIX systems. The stream file support is designed for efficient use in client/server applications.
- A hierarchical directory structure that allows objects to be organized like branches of a tree. To access an object, specify the path from the directories to the object.
- A common interface that allows users and applications to access stream files, database files, documents, and other objects that are stored on the iSeries server.

iSeries servers can support several different file systems with similar interfaces. A file system allows users and applications to access specific segments of storage that are organized as logical units. These logical units are files, directories, libraries, and objects.

For a list of iSeries file systems, see [Integrated File System Introduction](#).

For more information about the integrated file system, see [Database and File Systems](#).

The OS/400 file server can give clients access to all of the iSeries file systems or just QDLS, depending on the support provided by the client product.

The programs listed in the following table are included with this server.

File Server objects

| Program name | Library | Object type | Description |
|--------------|---------|-------------|---|
| QPWFSEVSO | QSYS | *PGM | Server program |
| QPWFSEVS2 | QSYS | *PGM | Server program |
| QPWFSEVSD | QSYS | *PGM | Daemon program |
| QPWFSEV | QSYS | *JOB | Job description used for server jobs |
| QPWFSEVSR | QSYS | *CLS | Class used for all file server and database server jobs |
| QPWFSEVSS | QSYS | *PGM | SSL server program |

Database server

The database server allows clients access to the functions included with DB2/400. This server provides:

- Support for remote SQL access
- Access to data through ODBC interfaces
- Database functions (such as creating and deleting files and adding and removing file members)
- Retrieval functions for obtaining information about database files that exist on the system (such as SQL catalog functions)

Additionally, you can use Distributed Relational Database Architecture (DRDA) with the database server. This topic provides information about using the following items with DRDA:

- SQL packages

- DRDA naming conventions
- DRDA rules and restrictions

For more information about DRDA, see Distributed database programming

The programs listed in the following table are included with this server.

Database server programs

| Program name | Library | Description |
|--------------|---------|-----------------------|
| QZDASOINIT | QSYS | Server program |
| QZDASON2 | QSYS | Sockets setup program |
| QZDASRVSD | QSYS | Daemon program |
| QZDASSINIT | QSYS | SSL server program |

Note: The *PGM objects QZDANDB, QZDAROI, QZDASQL, and QZDACMDP are used by the database server.

SQL packages: SQL packages bind SQL statements in an application program to a relational database. They are used to enhance the performance of applications that use dynamic SQL support by allowing the application to reuse information about the SQL requests. The database server is an application program that uses dynamic SQL requests. It supports the use of packages for frequently used SQL statements so that certain binding information can be reused.

For more information, see:

- SQL package names
- Cleaning up SQL packages

SQL package names: The database server can be used as a gateway to other relational databases that use DRDA. The database server automatically creates one or more SQL packages on the target relational database. The package names are generated according to the attributes currently used by the server.

<h8>Package names if the relational database is not an iSeries server

The package is created in a collection called QSQL400 on the application server if the relational database (RDB) is not an iSeries server. If the RDB is an iSeries server, the package is created in library QGPL. When the application server is not an iSeries server, the package name is QZD**abcde**, in which **abcde** corresponds to specific parser options being used. The following table shows the options for the package name.

Package name field options

| Field | Field description | Options |
|----------|-------------------|---|
| a | Date format | <ul style="list-style-type: none"> • ISO, JIS • USA • EUR • JUL |
| b | Time format | <ul style="list-style-type: none"> • JIS • USA • EUR, ISO |

| Field | Field description | Options |
|----------|--|---|
| c | Commitment control/ decimal delimiter | <ul style="list-style-type: none"> • *CS/period • *CS/comma • *CHG/period • *CHG/comma • *RR/period • *RR/comma |
| d | String delimiter | <ul style="list-style-type: none"> • apostrophe • quote |
| e | Maximum number of statements allowed for package | <ul style="list-style-type: none"> • 0 - 64 • 1 - 256 • 2 - 512 • 3 - 1024 |

Package names if the relational database is an iSeries server

When the application server is an iSeries server, the package name is QZDA**abcdef**, in which **abcdef** corresponds to specific parser options being used.

Package name field options

| Field | Field description | Options |
|----------|-----------------------------------|--|
| a | Date format | <ul style="list-style-type: none"> • ISO, JIS • USA • EUR • JUL • MDY • DMY • YMD |
| b | Time format and naming convention | <ul style="list-style-type: none"> • ISO, JIS and SQL naming • USA and SQL naming • EUR and SQL naming • HMS and SQL naming • ISO, JIS and system naming • USA and system naming • EUR and system naming • HMS and system naming |

| Field | Field description | Options |
|----------|--------------------------------|---|
| c | Commit level and decimal point | <ul style="list-style-type: none"> • *CS/period • *CS/comma • *ALL/period • *ALL/comma • *CHG/period • *CHG/comma • *NONE/period • *NONE/comma |
| d | String delimiter | <ul style="list-style-type: none"> • apostrophe • quote |
| e | Number of sections in package | <ul style="list-style-type: none"> • 0 - 64 • 1 - 256 • 2 - 512 • 3 - 1024 |
| f | Date and Time separation | <ul style="list-style-type: none"> • The high order bits of the character: • '1100'b - One of the ISO formats for da • '1101'b - Comma as date separation • '1110'b - Period as date separation • '1111'b - Colon as date separation • The low order bits of the character: • '0001'b - An ISO format of time • '0010'b - Comma as time separator • '0011'b - Period as time separator • '0100'b - Slash as time separator • '0101'b - Dash as time separator • '0110'b - Blank as time separator |

Cleaning up SQL packages: The packages used for DRDA functions are created automatically on your system as needed. You may want to periodically clean up these packages. To delete the packages, use the Delete SQL Package (DLTSQLPKG) command.

Delete the packages only if they are not used often. The package is created again if needed, but performance noticeably decreases when a package is created a second time.

Statement Naming Conventions: The following table provides a summary of the naming conventions enforced by the database server.

Statement Naming Conventions

| Statement | Dynamic SQL | Using an extended dynamic SQL package |
|-----------|--|--|
| Local | Statement name must adhere to iSeries naming convention, although the format of STMTxxxx is suggested | Statement name must adhere to iSeries naming convention, although the format of STMTxxxx is suggested |
| | Cursor name must adhere to iSeries naming conventions | Cursor name must adhere to iSeries naming conventions |
| DRDA | Statement name must be in the format of STMTxxxx | Statement name must be in the format of Sxxxx |
| | Cursor name must be in the format: CRSRyyyy for non-scrollable cursors or SCRSRyyyy for scrollable cursors where yyyy is the same as xxxx. | Cursor name must be in the format of Cyy for non-scrollable cursors where yy is the same as xxxx and yy is between 1 and 15. |

Notes:

1. The naming convention for statement names is not enforced on the local system, so a client application can share prepared statements with an iSeries application using the QSQPRCED system API.
2. The server appends a blank to the beginning of any statement name in the format of STMTxxxx. A host application must then append a leading blank to share statements with client applications that use the format STMTxxxx. The server does not append a leading blank if the statement name is not in the format of STMTxxxx.

Rules and restrictions when using DRDA: When using the database server as a gateway to other RDBs using DRDA, some limitations in functions must be followed.

The following table shows the functions that have limitations when you are connected to a remote system from the database server.

DRDA Functional Limits

| Function | Limitation |
|----------------------------------|--|
| Create package | Unsupported functions |
| Clear package | |
| Delete package | |
| Prepare | Enhanced prepare option not available when using DRDA. |
| Extended dynamic package support | <ul style="list-style-type: none"> • Only available when connected to an iSeries server running OS/400 v2r3 or later • Can only access statements in a package using the naming convention of 'STMTxxxx' in which xxxx is the section number |
| Describe parameter markers | Only available when connected to an iSeries server. |
| Commit hold | Only valid if connected to an iSeries server |
| Commit level *NONE | Not supported |
| Commit level *CHANGE | Only supported if the target RDB is an iSeries. All other RDBs require a *CS or *ALL commit level. |

Data Queue Server

A data queue is an object that is used by iSeries application programs for communications. Applications can use data queues to pass data between jobs. Multiple iSeries jobs can send or receive data from a single data queue.

iSeries Access for Windows provides APIs that can allow PC applications to work with iSeries data queues with the same ease that iSeries applications can. This extends iSeries application communications to include processes running on a remote PC.

The programs listed in the following table are included with this server.

Data Queue server program provided for use with sockets support

| Program name | Library | Description |
|--------------|---------|----------------|
| QZHQSSRV | QSYS | Server program |
| QZHQSRVD | QSYS | Daemon program |

Network Print Server

The OS/400 network print server allows enhanced client control over print resources on the iSeries server. This print server provides the following capabilities to each client by requesting print serving:

Spooled file

Create, seek, open, read, write, close, hold, release, delete, move, send, call exit program, change attributes, retrieve message, answer message, retrieve attributes, and list

Writer job

Start, end, and list

Printer device

Retrieve attributes and list

Output queue

Hold, release, purge, list, and retrieve attributes

Library

List

Printer file

Retrieve attributes, change attributes, and list

Network print server

Change attributes and retrieve attributes

The programs listed in the following table are included with this server.

Network Print server

| Program name | Library | Description |
|--------------|---------|----------------|
| QNPSEVS | QSYS | Server program |
| QNPSEVD | QSYS | Daemon program |

Central Server

The central server provides the following services for clients:

- License management

The initial request from either Data Transfer or PC5250 reserves a license for that iSeries Access for Windows user. The server remains active until the release delay timeout expires. The license will be held until it is released or the server job is ended. To see which licenses are reserved, use iSeries Navigator to view the iSeries system's properties.

- Retrieve conversion map

The central server retrieves conversion maps for clients who need them. These conversion maps are usually used for ASCII to EBCDIC conversions and for EBCDIC to ASCII conversions. The client can request a map by giving the correct source, the target coded character set identifier (CCSID), and a table of code points to be converted. The server then returns the correct mapping for the client to use.

The programs listed in the following table are included with this server.

Central server programs

| Program name | Library | Description |
|--------------|---------|----------------|
| QZCSRVS | QSYS | Server program |
| QZCSRVSD | QSYS | Daemon program |

Remote Command and Distributed Program Call Server

The remote command and distributed program call server allows users and applications to issue iSeries CL commands and call programs.

The remote command allows the user to run multiple commands in the same job. It also offers a better security check for iSeries users with limited capabilities (LMTCPB =*YES) in their user profile.

The distributed program call support allows applications to call iSeries programs and pass parameters (input and output). After the program runs on the iSeries server, the output parameter values return to the client application. This process allows applications to access iSeries resources easily without worry about the communications and conversions that must take place.

The programs listed in the following table are included with this server.

Remote Command and Distributed Program Call server programs

| Program name | Library | Description |
|--------------|---------|----------------|
| QZRCSRVS | QSYS | Server program |
| QZRCSRVSD | QSYS | Daemon program |

Signon Server

The Signon server provides security for clients. This security function prevents access to the system by users with expired passwords, validates user profile passwords and returns user profile security information for use with password caching and iSeries Navigator Application Administration.

The programs listed in the following table are included with this server.

Signon server programs

| Program name | Library | Description |
|--------------|---------|----------------|
| QZSOSIGN | QSYS | Server program |
| QZSOSGND | QSYS | Daemon program |

Server Port Mapper

The port mapper provides a way for the client to find the port for a particular service (server). The port mapper finds the ports in the TCP/IP Service Table.

The program listed in the following table is included with this server.

Server port mapper

| Program name | Library | Description |
|--------------|---------|----------------------------|
| QZSOSMAPD | QSYS | Server port mapper program |

Using OS/400 host servers

This topic describes how to manage the OS/400 server jobs. It describes the subsystems in which the servers run, the objects that affect the servers, and how to manage these resources.

The servers shipped with the OS/400 program do not typically require any changes to your existing system configuration in order to work correctly. They are set up and configured when you install OS/400. You may want to change the way the system manages the server jobs to meet your needs, solve problems, improve system performance, or simply view the jobs on the system. To make such changes and meet processing requirements, you must know which objects affect which pieces of the system and how to change those objects. To really understand how to manage your system, refer to Work Management before you continue with this chapter.

Establishing client/server communication

Learn the process for starting and ending communication between clients and host servers. This topic also includes each server's port number, and a description of server daemons and their role in communication.

Subsystems on OS/400

This topic presents descriptions of the subsystems on OS/400, and illustrates how to autostart and prestart jobs.

System values on the iSeries

Lists and describes system values important in client/server environments.

Identifying server jobs on the iSeries

Shows how to display server jobs using either iSeries Navigator or the green screen.

Using EZ Setup and iSeries Navigator with host servers

Describes how to tell if the required communication path is active, and how to start it if necessary.

Establishing client/server communications

Client/Server communication is established in the following steps:

1. To initiate a server job that uses sockets communications support, the client system connects to a particular server's port number.
2. A server daemon must be started (with the STRHOSTSVR command) to listen for and accept the client's connection request. Upon accepting the connection request, the server daemon issues an internal request to attach the client's connection to a server job.
3. This server job may be a prestarted job or, if prestart jobs are not used, a batch job that is submitted when the client connection request is processed. The server job handles any further communications with the client. The initial data exchange includes a request that identifies the user profile and password that are associated with the client user.
4. Once the user profile and password are validated, the server job switches to this user profile and changes the job by using many of the attributes defined for the user profile, such as accounting code and output queue.

For more information, see:

- Port numbers for host servers
- Starting host servers
- Ending host servers

Server to Client Communications

iSeries Access for Windows uses TCP/IP to communicate with the iSeries system servers. The optimized servers use OS/400 sockets support to communicate with clients. The OS/400 sockets support is compatible with Berkeley Software Distributions 4.3 sockets over TCP/IP. Sockets support is provided with the 5769-TC1 product that is installed on the iSeries server.

See the TCP/IP Configuration and Reference manual for more information about communications.

Host Servers port numbers: Each type of server has its own server daemon, which listens on a port for incoming client connection requests. There are exceptions to this. For instance, the transfer function over sockets uses the database server daemon; the network drive server uses the file server daemon; and the virtual print server uses the network print server daemon. In addition, the server mapper daemon also listens on a specified port, and allows a client to obtain the current port number for a specified server.

Each of the server daemons listen on the port number that is provided in the service table for the specified service name. For example, the network print server daemon, with the initial configuration that is provided, listens on port number 8474, which is associated with service name 'as-netprt.' The server mapper daemon listens on the well-known port. The well-known server mapper port number is 449. The well-known port number is reserved for the exclusive use of the OS/400 Host Servers. Therefore, the entry for the 'as-svrmap' service name should not be removed from the service table.

The port numbers for each server daemon are not fixed; the service table can be modified by using different port numbers if your installation requires such changes. You can change where the port number is retrieved from the iSeries Navigator system properties connection tab. However, the service name must remain the same as that shown in following tables. Otherwise, the server daemons cannot establish a port number on which to accept incoming requests for client connection.

If a new service table entry is added to identify a different port number for a service, any pre-existing service table entries for that service name should be removed. Removing these entries eliminates the duplication of the service name in the table and eliminates the possibility of unpredictable results when the server daemon starts.

Port numbers for host servers and server mapper

View each server's port number for the optimized servers and server mapper that use sockets over TCP communication support and those that use Secure Sockets Layer (SSL).

Starting host servers: To start the OS/400 Host Servers, use the STRHOSTSVR CL command. This command starts the host server daemons and the server mapper daemon. It also attempts to start the prestart job associated with that server.

Note:

You can use iSeries Navigator to configure your system so that servers start automatically when you start Transmission Control Protocol (TCP) with the STRTCP command. Newly shipped systems will do this by default.

Each host server type has a server daemon. There is a single server mapper daemon for the system. The client PC application uses the port number to connect to the host server daemon. The server daemon accepts the incoming connection request and routes it to the server job for processing.

STRHOSTSVR command values:

SERVER

***ALL** Starts all host server daemons and the server mapper daemon.

***CENTRAL**

Starts the central server daemon in QSYSWRK subsystem. The daemon job is QZSCSRVSD, and the server prestart job is QZSCSRVS.

***DATABASE**

Starts the database server daemon in QSERVER subsystem. The daemon job is QZDASRVSD, and the associated server prestart jobs are QZDASOINIT, QZDASSINIT, and QTFPJTCP.

***DTAQ**

Starts the data queue server daemon in QSYSWRK subsystem. The daemon job is QZHQSRVD, and the associated server prestart job is QZHQSSRV.

***FILE**

Starts the file server daemon in QSERVER subsystem. The daemon job is QPWFSESRVSD, and the associated server prestart jobs are QPWFSESRVSO, QPWFSESRVSS, and QPWFSESRV2.

***NETPRT**

Starts the network print server daemon in QSYSWRK subsystem. The daemon job is QNPSESRVSD, and the associated server prestart jobs are QNPSESRVS and QIWVPPJT.

***RMTCMD**

Starts the remote command and the distributed program call server daemon in QSYSWRK subsystem. The daemon job is QZRCSRVD, and the associated server prestart job is QZRCSRVS.

***SIGNON**

Starts the signon server daemon in QSYSWRK subsystem. The daemon job is QZSOSGND and the associated server prestart job QZSOSIGN.

***SVRMAP**

Starts the server mapper daemon in QSYSWRK subsystem. The daemon job is QZSOSMAPD.

Note:

If the daemon job runs in the QSYSWRK directory, the associated server prestart jobs will run in the QUSRWRK directory by default. Additionally, database server prestart jobs will run in QUSRWRK subsystem by default.

Optional parameter:

RQDPCL

Specifies which communication protocols are required to be active for the host server daemons to start.

Single Values:

***ANY** The TCP/IP communication protocol must be active at the time the STRHOSTSVR command is issued. If TCP/IP is not active, escape message PWS300D will be issued and the host server daemons will not be started. A diagnostic message (PWS3008) will also be issued, if TCP/IP is found to be inactive.

***NONE**

No communication protocols need to be active at the time the STRHOSTSVR command is issued for the host server daemons to start. No messages will be issued for protocols which are inactive.

***TCP** The TCP/IP communication protocol must be active at the time the STRHOSTSVR command is issued. If TCP/IP is not active, diagnostic message PWS3008 and escape message PWS300D will be issued and the host server daemons will not be started.

Here are some STRHOSTSVR

Example: STRHOSTSVR: **Example 1: Starting all host server daemons**

STRHOSTSVR(*ALL)

This command starts all the server daemons and the server mapper daemon, as long as at least one communication protocol is active.

Example 2: To start specific server daemons

STRHOSTSVR SERVER(*CENTRAL *SVRMAP) RQDPCL(*NONE)

This command starts the central server daemon and the server mapper daemon in QSYSWRK subsystem, even if no communication protocols are active.

Example 3: Specification of one required protocol:

STRHOSTSVR SERVER(*ALL) RQDPCL(*TCP)

This command starts all the host server daemons and the server mapper daemon in QSYSWRK subsystem, as long as TCP/IP is active.

Ending host servers: To end the OS/400 Host servers, use the ENDMETHODSVR CL command. This command ends the host server daemons and the server mapper daemon. If a server daemon ends while servers of that type are connected to client applications, the server jobs remain active until communication with the client application ends, unless the optional ENDMETHODCNN parameter is specified. Subsequent connection requests from the client application to that server fail until the server daemon starts again.

If the server mapper daemon ends, any existing client connections to the server jobs are unaffected. Subsequent requests from a client application to connect to the server mapper fail until the server mapper starts again.

The ENDMETHODCNN parameter may be specified in order to end active connections to the *DATABASE and *FILE servers. This will cause the server jobs that are servicing these connections to end. The active connections can only be ended if the corresponding daemon job is also being ended. If the *DATABASE keyword is specified, the QZDASOINIT and QZDASSINIT jobs with active connections will be ended. If the *FILE keyword is specified, the QPWFSERVSO and QPWFSERVSS jobs with active connections will be ended.

Note:

If you use the ENDMETHODSVR command to end a particular daemon that is not active, you get a diagnostic message. Use ENDMETHODSVR SERVER(*ALL) if you want to end any active daemons. You do not see a diagnostic message with the *ALL value.

ENDMETHODSVR command values:

SERVER

***ALL** Ends the server daemons and the server mapper daemon if active. If used, the system allows no other special values.

***CENTRAL**

Ends the central server daemon in QSYSWRK subsystem.

***DATABASE**

Ends the database server daemon in QSERVER subsystem.

***DTAQ**

Ends the data queue server daemon in QSYSWRK subsystem.

***FILE**

Ends the file server daemon in QSERVER subsystem.

***NETPRT**

Ends the network print server daemon in QSYSWRK subsystem.

***RMTCMD**

Ends the remote command and the distributed program call server daemon in QSYSWRK subsystem.

***SIGNON**

Ends the signon server daemon in QSYSWRK subsystem.

***SVRMAP**

Ends the server mapper daemon in QSYSWRK subsystem.

Optional Parameter**ENDACTCNN**

Specifies whether the active connections for the specified servers will be ended.

Single Value***NONE**

No active connections will be ended.

Specific Server Values***DATABASE**

The active connections being serviced by the QZDASOINIT and QZDASSINIT server jobs will be ended. The server jobs that are servicing these connections will also be ended.

***FILE** The active connections being serviced by the QPWFSESRVO and QPWFSESRVSS server jobs will be ended. The server jobs servicing these connections will also be ended.

Here are some ENDDHOSTSVR examples.

Example: ENDDHOSTSVR: **Example 1: Ending all host server daemons**

```
ENDDHOSTSVR SERVER(*ALL)
```

This command ends all the server daemons and the server mapper daemon.

Example 2: To end specific server daemons

```
ENDDHOSTSVR SERVER(*CENTRAL *SVRMAP)
```

End the central server daemon and the server mapper daemon.

Example 3: Ending specific server daemons and active connections

```
ENDDHOSTSVR SERVER(*CENTRAL *DATABASE) ENDDACTCNN(*DATABASE)
```

This command ends the central server daemon in the QSYSWRK subsystem and the database server daemon in the QSERVER subsystem. Additionally, the active connections to the *DATABASE server, and the QZDASOINIT and QZDASSINIT server jobs that are servicing these connections will end.

Subsystems on the iSeries server

The following subsections describe which system-supplied subsystems are used for each of the server functions. These sections also detail how the subsystem descriptions relate to the server jobs.

A subsystem description defines how, where, and how much work enters a subsystem, and which resources the subsystem uses to do the work.

Subsystems used for server jobs

Use of autostart jobs

Autostart jobs perform one-time initialization or do repetitive work that is associated with a particular subsystem. The autostart jobs associated with a particular subsystem are automatically started each time the subsystem is started.

Use of prestart jobs

Subsystems Used for Server Jobs: The server jobs are configured to run in different subsystems, depending on their function. The following are the subsystems used for the server jobs.

QSYSWRK

All of the daemon jobs (with the exception of the file server daemon job and the database server daemon job) run in this subsystem. The file server and database server daemon jobs run in the QSERVER subsystem.

QUSRWRK

This subsystem is where the server jobs run for these servers:

- Network Print
- Remote Command/Program Call
- Central
- Data Queue
- Signon
- Database

QSERVER

The file server, its associated daemon job and the database server daemon job must run in this subsystem.

If this subsystem is not active, requests to establish a connection to the file server or the database server will fail.

Automatically Starting Subsystems

The QSYSWRK subsystem starts automatically when you IPL, regardless of the value specified for the controlling subsystem.

If you use the default startup program provided with the system, the QSERVER and QUSRWRK subsystems start automatically when you IPL. The system startup program is defined in the QSTRUPPGM system value, and the default value is QSTRUP QSYS.

If you want to change the system startup, you can change the QSTRUPPGM system value to call your own program. You can use the shipped program QSTRUP in QSYS as a base for the start-up program that you create.

Note: If you use the database server or file server and you made changes to the system startup, you must ensure that the startup program starts the QSERVER subsystem.

Beginning in V5R1, TCP/IP is automatically started by the system without requiring a change to the system startup program. The host servers are automatically started when TCP/IP is started. When TCP/IP is started, it ensures QUSRWRK and QSERVER are started before starting the host servers.

If slip installing V5R1 (or later) on a system that was at a release prior to V5R1, and if the startup program used by the system had been changed to start TCP/IP, then the system will automatically start TCP/IP, and the startup program's attempt will fail.

The IPL attribute, STRTCP, can force the system to not automatically start TCP/IP at IPL. It is recommended to leave this value at the shipped setting of *YES, (start TCP/IP) but the option is available if necessary.

Use of Autostart Jobs: The QSERVER subsystem has an autostart job defined for the file server and database server jobs. If this job is not running, the servers cannot start. The subsystem will not end when the job disappears. If a problem occurs with this job, you may want to end and restart the QSERVER subsystem.

The QSYSWRK subsystem has an autostart job defined for all of the optimized servers. This job monitors for events sent when a STRTCP command has been issued. This way, the server daemon jobs can dynamically determine when TCP/IP has become active. The daemon jobs then begin to listen on the appropriate ports. If the autostart job is not active, and TCP/IP is started while the host servers are active, the following sequence of commands must be issued in order to start using TCP/IP:

1. ENHOSTSVR *ALL
2. STRHOSTSVR *ALL

The autostart job is named QZBSEVTM. If the job is not active, it can be started by issuing the following command:

```
QSYS/SBMJOB CMD(QSYS/CALL PGM(QSYS/QZBSEVTM)) JOB(QZBSEVTM) JOBID(QSYS/QZBSEJBD)
PRTDEV(*USRPRF) OUTQ(*USRPRF) USER(QUSER) PRTTXT(*SYSVAL) SYSLIBL(*SYSVAL)
CURLIB(*CRTDFT) INLLIBL(*JOBID) SRTSEQ (*SYSVAL) LANGID(*SYSVAL) CNTRYID(*SYSVAL)
CCSID(*SYSVAL)
```

Note: Only one instance of program QZBSEVTM can be running at any one time.

Use of Prestart Jobs: A prestart job is a batch job that starts running before a program on a remote system initiates communications with the server. Prestart jobs use prestart job entries in the subsystem description to determine which program, class, and storage pool to use when the jobs are started. Within a prestart job entry, you must specify attributes for the subsystem to use to create and to manage a pool of prestart jobs.

Prestart jobs increase performance when you initiate a connection to a server. Prestart job entries are defined within a subsystem. Prestart jobs become active when that subsystem is started, or they can be controlled with the Start Prestart Job (STRPJ) and End Prestart Job (ENDPJ) commands.

System information that pertains to prestart jobs (such as DSPACTPJ) uses the term 'program start request' exclusively to indicate requests made to start prestart jobs, even though the information may pertain to a prestart job that was started as a result of a sockets connection request.

Notes:

- Prestart jobs can be reused, but there is no automatic cleanup for the prestart job once it has been used and subsequently returned to the pool. The number of times the prestart job is reused is determined by the value specified for the maximum number of uses (MAXUSE) value of the ADDPJE or CHGPJE CL commands. This means that resources that are used by one user of the prestart job must be cleaned up before ending use of the prestart job. Otherwise, these resources

will maintain the same status for the next user that uses the prestart job. For example, a file that is opened but never closed by one user of a prestart job remains open and available to the following user of the same prestart job.

- By default, some of the server jobs run in QUSRWRK or QSERVER. Using iSeries Navigator, you can configure some or all of these servers to run in a subsystem of your choice.
 1. Double-click **iSeries Navigator** → **Network** → **Servers** → **iSeries Access**.
 2. Right-click the server that you want to configure subsystems for and select **Properties**.
 3. Configure the server using the Subsystems page.

If you move jobs from the default subsystem, you must:

1. Create your own subsystem description.
2. Add your own pre-start jobs using the ADDPJE command. Set the STRJOBS parameter to *YES.

If you do not do this, your jobs will run in the default subsystem.

All of the OS/400 servers that are supported by the sockets communications interface support prestart jobs.

These servers are:

Network print server
 Remote command and distributed program call server Central server
 Database server
 Secure Database server
 File server
 Secure File server
 Data Queue server
 Signon server (unique to servers using sockets communications support)

The following lists provide each of the prestart job entry attributes, and the initial values that are configured for the host servers using sockets communications support.

Subsystem Description

The subsystem that contains the prestart job entries.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | QUSRWRK |
| Remote CMD/PGM Call | QUSRWRK |
| Central | QUSRWRK |
| Database | QUSRWRK |
| Secure Database | QUSRWRK |
| File | QSERVER |
| Secure File | QSERVER |
| Data Queue | QUSRWRK |
| Signon | QUSRWRK |

Program library/name

The program that is called when the prestart job is started.

| OS/400 Server | Value |
|----------------------|-----------------|
| Network Print | QSYS/QNPSEVS |
| Remote CMD/PGM Call | QSYS/QZRCSEVS |
| Central | QSYS/QZSCSEVS |
| Database | QSYS/QZDASOINIT |
| Secure Database | QSYS/QZDASSINIT |

OS/400 Server

File
 Secure File
 Data Queue
 Signon

Value

QSYS/QPWFSEVS0
 QSYS/QPWFSEVS5
 QSYS/QZHQSSRV
 QSYS/QZSOSIGN

User profile

The user profile that the job runs under. This is what the job shows as the user profile. When a request to start a server is received from a client, the prestart job function switches to the user profile that is received in that request.

OS/400 Server

Network Print
 Remote CMD/PGM Call
 Central
 Database
 Secure Database
 File
 Secure File
 Data Queue
 Signon

Value

QUSER
 QUSER
 QUSER
 QUSER
 QUSER
 QUSER
 QUSER
 QUSER
 QUSER

Job name

The name of the job when it is started.

OS/400 Server

Network Print
 Remote CMD/PGM Call
 Central
 Database
 Secure Database
 File
 Secure File
 Data Queue
 Signon

Value

*PGM
 *PGM
 *PGM
 *PGM
 *PGM
 *PGM
 *PGM
 *PGM
 *PGM

Job description

The job description used for the prestart job. Note that if *USRPRF is specified, the job description for the profile that this job runs under will be used. This means QUSER's job description will be used. Some attributes from the requesting user's job description are also used; for example, print device and output queue are swapped from the requesting user's job description.

OS/400 Server

Network Print
 Remote CMD/PGM Call
 Central
 Database
 Secure Database
 File
 Secure File
 Data Queue
 Signon

Value

QSYS/QZBSJOB
 QSYS/QZBSJOB
 QSYS/QZBSJOB
 *USRPRF
 *USRPRF
 *USRPRF
 *USRPRF
 QSYS/QZBSJOB
 QSYS/QZBSJOB

Start jobs

Indicates whether prestart jobs are to automatically start when the subsystem is started. These prestart job entries are shipped with a start jobs value of *YES to ensure that the server jobs are available. The STRHOSTSVR command starts each prestart job as part of its processing.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | *YES |
| Remote CMD/PGM Call | *YES |
| Central | *YES |
| Database | *YES |
| Secure Database | *YES |
| File | *YES |
| Secure File | *YES |
| Data Queue | *YES |
| Signon | *YES |

Initial number of jobs

The number of jobs that are started when the subsystem starts. This value is adjustable to suit your particular environment and needs.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | 1 |
| Remote CMD/PGM Call | 1 |
| Central | 1 |
| Database | 1 |
| Secure Database | 1 |
| File | 1 |
| Secure File | 1 |
| Data Queue | 1 |
| Signon | 1 |

Threshold

The minimum number of available prestart jobs for a prestart job entry. When this threshold is reached, additional prestart jobs automatically start. Threshold maintains a certain number of jobs in the pool.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | 1 |
| Remote CMD/PGM Call | 1 |
| Central | 1 |
| Database | 1 |
| Secure Database | 1 |
| File | 1 |
| Secure File | 1 |
| Data Queue | 1 |
| Signon | 1 |

Additional number of jobs

The number of additional prestart jobs that are started when the threshold is reached.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | 2 |
| Remote CMD/PGM Call | 2 |
| Central | 2 |
| Database | 2 |

| OS/400 Server | Value |
|----------------------|--------------|
| Secure Database | 2 |
| File | 2 |
| Secure File | 2 |
| Data Queue | 2 |
| Signon | 2 |

Maximum number of jobs

The maximum number of prestart jobs that can be active for this entry.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | *NOMAX |
| Remote CMD/PGM Call | *NOMAX |
| Central | *NOMAX |
| Database | *NOMAX |
| Secure Database | *NOMAX |
| File | *NOMAX |
| Secure File | *NOMAX |
| Data Queue | *NOMAX |
| Signon | *NOMAX |

Maximum number of uses

The maximum number of uses of the job. A value of 200 indicates that the prestart job will end after 200 requests to start the server have been processed.

Note: The database server does not reuse any of the prestart jobs, even if this value is set to a value greater than one.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | 200 |
| Remote CMD/PGM Call | 1 |
| Central | 200 |
| Database | 1 |
| Secure Database | 200 |
| File | *NOMAX |
| Secure File | *NOMAX |
| Data Queue | 200 |
| Signon | 200 |

Wait for job

This causes a client connection request to wait for an available server job if the maximum number of jobs has been reached.

| OS/400 Server | Value |
|----------------------|--------------|
| Network Print | *YES |
| Remote CMD/PGM Call | *YES |
| Central | *YES |
| Database | *YES |
| Secure Database | *YES |
| File | *YES |
| Secure File | *YES |
| Data Queue | *YES |
| Signon | *YES |

Pool identifier

The subsystem pool identifier in which this prestart job runs.

| OS/400 Server | Value |
|---------------------|-------|
| Network Print | 1 |
| Remote CMD/PGM Call | 1 |
| Central | 1 |
| Database | 1 |
| Secure Database | 1 |
| File | 1 |
| Secure File | 1 |
| Data Queue | 1 |
| Signon | 1 |

Class

The name and library of the class the prestart job runs under.

| OS/400 Server | Value |
|---------------------|----------------|
| Network Print | QGPL/QCASERV |
| Remote CMD/PGM Call | QGPL/QCASERV |
| Central | QGPL/QCASERV |
| Database | QSYS/QPWFSEVER |
| Secure Database | QSYS/QPWFSEVER |
| File | QSYS/QPWFSEVER |
| Secure File | QSYS/QPWFSEVER |
| Data Queue | QGPL/QCASERV |
| Signon | QGPL/QCASERV |

When the start jobs value for the prestart job entry has been set to *YES, and the remaining values are at their initial settings, the following actions take place for each prestart job entry:

- When the subsystem is started, one prestart job for each server is started.
- When the first client connection request processes for a specific server, the initial job is used and the threshold is exceeded.
- Additional jobs are started for that server based on the number that is defined in the prestart job entry.
- The number of available jobs is always at least one.
- The subsystem periodically checks the number of prestart jobs that are ready to process requests, and ends excess jobs. The subsystem always leaves at least the number of prestart jobs specified in the initial jobs parameter.

Monitoring Prestart Jobs

Use the Display Active Prestart Jobs (DSPACTPJ) command to monitor the prestart jobs. For example, to monitor prestart jobs for the signon server, you must know the subsystem your prestart jobs are in (QUSRWRK or a user-defined subsystem) and the program (for example, QZSOSIGN).

The DSPACTPJ command provides the following information:

```
+-----+
|                Display Active Prestart Jobs                AS400597
|                01/12/95 16:39:25
| Subsystem . . . . . : QUSRWRK   Reset date . . . . . : 01/11/95
| Program . . . . . : QZSOSIGN   Reset time . . . . . : 16:54:50
| Library . . . . . : QSYS      Elapsed time . . . . . : 0023:12:21
|
| Prestart jobs:
| Current number . . . . . : 10
+-----+
```

```

Average number . . . . . : 8.5
Peak number . . . . . : 25

Prestart jobs in use:
Current number . . . . . : 5
Average number . . . . . : 4.3
Peak number . . . . . : 25

More...

-----+-----+-----+
Subsystem . . . . . : QUSRWRK      Reset date . . . . . : 01/12/95 16:39:25
Program . . . . . : QZSOSIGN     Reset time . . . . . : 01/11/95 16:54:50
Library . . . . . : QSYS        Elapsed time . . . . . : 0023:12:21

Program start requests:
Current number waiting . . . . . : 0
Average number waiting . . . . . : .2
Peak number waiting . . . . . : 4
Average wait time . . . . . : 00:00:20.0
Number accepted . . . . . : 0
Number rejected . . . . . : 0

Bottom

Press Enter to continue.

F3=Exit  F5=Refresh  F12=Cancel  F13=Reset statistics

```

Managing Prestart Jobs

Pressing the **(F5)** key while on the Display Active Prestart Jobs display can refresh the information presented for an active prestart job. The information about program start requests can indicate whether you need to change the available number of prestart jobs. If the information indicates that program start requests are waiting for an available prestart job, you can change prestart jobs with the Change Prestart Job Entry (CHGPJE) command.

If the program start requests are not acted on quickly, you can do any combination of the following:

- Increase the threshold
- Increase the parameter value for the initial number of jobs (INLJOBS)
- Increase the parameter value for the additional number of jobs (ADLJOBS)

The key is to ensure an available prestart job exists for every request.

Removing Prestart Job Entries

If you decide that you do not want the servers to use the prestart job function, you must do the following:

1. End the prestarted jobs with the End Prestart Job (ENDPJ) command.

Prestarted jobs ended with the ENDPJ command are started the next time the subsystem is started if start jobs *YES is specified in the prestart job entry or when the STRHOSTSVR command is issued for the specified server type. If you only end the prestart job and don't take the next step, any requests to start the particular server will fail.

2. Remove the prestart job entries in the subsystem description with the Remove Prestart Job Entry (RMVPJE) command.

The prestart job entries that are removed with the RMVPJE command are permanently removed from the subsystem description. Once the entry is removed, new requests for the server will succeed.

Routing Entries

When a daemon job is routed to a subsystem, the job is using the routing entries in the subsystem description. The routing entries for the host server daemon jobs are added to the subsystem description when the STRHOSTSVR command is issued. These jobs are started under the QUSER user profile. For daemon jobs that are submitted to the QSYSWRK subsystem, the QSYSNOMAX job queue is used. For daemon jobs that are submitted to the QSERVER subsystem, the QPWFSERVER job queue is used.

The server jobs run in the same subsystem as their corresponding daemon job. The characteristics of the server jobs are taken from their prestart job entry. If prestart jobs are not used for the servers, then the server jobs start with the characteristics of their corresponding daemon job.

The following information provides the initial configuration in the IBM-supplied subsystems for each of the server daemon jobs.

Network Print Server daemon

| | |
|-----------------|--------------|
| Subsystem | QSYS/QSYSWRK |
| Job Queue | QSYSNOMAX |
| User | QUSER |
| Routing Data | QNPSEVRD |
| Job Name | QNPSEVRD |
| Class | QGPL/QCASERV |
| Sequence Number | 2538 |

Remote Cmd/Pgm Call Server daemon

| | |
|-----------------|--------------|
| Subsystem | QSYS/QSYSWRK |
| Job Queue | QSYSNOMAX |
| User | QUSER |
| Routing Data | QZRCSRVD |
| Job Name | QZRCSRVD |
| Class | QGPL/QCASERV |
| Sequence Number | 2539 |

Central Server daemon

| | |
|-----------------|--------------|
| Subsystem | QSYS/QSYSWRK |
| Job Queue | QSYSNOMAX |
| User | QUSER |
| Routing Data | QZSCSRVD |
| Job Name | QZSCSRVD |
| Class | QGPL/QCASERV |
| Sequence Number | 2536 |

Database Server daemon

| | |
|-----------------|----------------|
| Subsystem | QSYS/QSERVER |
| Job Queue | QPWFSEVER |
| User | QUSER |
| Routing Data | QZDASRVSD |
| Job Name | QZDASRVSD |
| Class | QSYS/QPWFSEVER |
| Sequence Number | 600 |

File Server daemon

| | |
|-----------------|----------------|
| Subsystem | QSYS/QSERVER |
| Job Queue | QPWFSEVER |
| User | QUSER |
| Routing Data | QPWFSEVERSD |
| Job Name | QPWFSEVERSD |
| Class | QSYS/QPWFSEVER |
| Sequence Number | 200 |

Data Queue Server daemon

| | |
|-----------------|---------------|
| Subsystem | QSYS/QSYSWRK |
| Job Queue | QSYSNOMAX |
| User | QUSER |
| Routing Data | QZHQSRVD |
| Job Name | QZHQSRVD |
| Class | QGPL/QCASERVR |
| Sequence Number | 2537 |

Signon Server daemon

| | |
|-----------------|---------------|
| Subsystem | QSYS/QSYSWRK |
| Job Queue | QSYSNOMAX |
| User | QUSER |
| Routing Data | QZSOSGND |
| Job Name | QZSOSGND |
| Class | QGPL/QCASERVR |
| Sequence Number | 2540 |

Server Mapper daemon

| | |
|-----------------|---------------|
| Subsystem | QSYS/QSYSWRK |
| Job Queue | QSYSNOMAX |
| User | QUSER |
| Routing Data | QZSOSMAPD |
| Job Name | QZSOSMAPD |
| Class | QGPL/QCASERVR |
| Sequence Number | 2541 |

System Values on the iSeries server

A system value contains control information that operates certain parts of the system. A user can change the system values to define the work environment. Examples of system values are system date and library list.

The iSeries server has many system values. The following values are of particular interest in a client/server environment.

QAUDCTL

Audit control. This system value contains the on and off switches for object and user level auditing. Changes that are made to this system value take effect immediately.

QAUDENDACN

Audit journal error action. This system value specifies the action the system takes if errors occur when an audit journal entry is being sent by the operating system security audit journal. Changes that are made to this system value take effect immediately.

QAUDFRCLVL

Force audit journal. This system value specifies the number of audit journal entries that can be written to the security auditing journal before the journal entry data is forced to auxiliary storage. Changes that are made to this system value take effect immediately.

QAUDLVL

Security auditing level. Changes made to this system value take effect immediately for all jobs running on the system.

QAUTOVRT

Determines whether the system should automatically create virtual devices. This is used with display station pass-through and Telnet sessions.

QCCSID

The coded character set identifier, which identifies:

- A specific set of encoding scheme identifiers
- Character set identifiers
- Code page identifiers
- Additional coding-related information that uniquely identifies the coded graphic character representation needed by the system

This value is based on the language that is installed on the system. It determines whether data must be converted to a different format before being presented to the user. The default value is 65535, which means this data is not converted.

QCTLSBSD

The controlling subsystem description

QDSPSGNINF

Determines whether the sign-on information display shows after sign-on by using the 5250 emulation functions (workstation function, PC5250).

QLANGID

The default language identifier for the system. It determines the default CCSID for a user's job if the job CCSID is 65535. The clients and servers use this default job CCSID value to determine the correct conversion for data that is exchanged between the client and the server.

QLMTSECOFR

Controls whether a user with all-object (*ALLOBJ) or service (*SERVICE) special authority can use any device. If this value is set to 1, all users with *ALLOBJ or *SERVICE special authorities must have specific *CHANGE authority to use the device.

This affects virtual devices for 5250 emulation. The shipped value for this is 1. If you want authorized users to sign-on to PCs, you must either give them specific authority to the device and controller that the PC uses or change this value to 0.

QMAXSIGN

Controls the number of consecutive incorrect sign-on attempts by local and remote users. Once the QMAXSIGN value is reached, the system determines the action with the QMAXSGNACN system value.

If the QMAXSGNACN value is 1 (vary off device), the QMAXSIGN value does not affect a user who enters an incorrect password on the PC when they are starting the connection.

This is a potential security exposure for PC users. The QMAXSGNACN should be set to either 2 or 3.

QMAXSGNACN

Determines what the system does when the maximum number of sign-on attempts is reached at any device. You can specify 1 (vary off device), 2 (disable the user profile) or 3 (vary off device and disable the user profile). The shipped value is 3.

QPWDEXPITV

The number of days for which a password is valid. Changes that are made to this system value take effect immediately.

QPWDLMTAJC

Limits the use of adjacent numbers in a password. Changes that are made to this system value take effect the next time a password is changed.

QPWDLMTCHR

Limits the use of certain characters in a password. Changes that are made to this system value take effect the next time a password is changed.

QPWDLMTREP

Limits the use of repeating characters in a password. Changes that are made to this system value take effect the next time a password is changed.

QPWDLVL

Determines the level of password support for the system, which includes the password length that the iSeries server will support, the type of encryption used for passwords, and whether AS/400 NetServer passwords for Windows 95/98/ME clients will be removed from the system. Changes that to this system value take effect on the next IPL.

Warning! If you set this value to support long passwords, you must upgrade all client PCs for long password support (Express V5R1) before setting this value. Otherwise, all pre-V5R1 clients will be unable to log onto the iSeries server.

QPWDMAXLEN

The maximum number of characters in a password. Changes that are made to this system value take effect the next time a password is changed.

QPWDMINLEN

The minimum number of characters in a password. Changes that are made to this system value take effect the next time a password is changed.

QPWDPOSDIF

Controls the position of characters in a new password. Changes that are made to this system value take effect the next time a password is changed.

QPWDRQDDGT

Requires a number in a new password. Changes that are made to this system value take effect the next time a password is changed.

QPWDRQDDIF

Controls whether the password must be different than previous passwords.

QPWDVLDPGM

Password validation program name and library that are supplied by the computer system. Both an object name and library name can be specified. Changes that are made to this system value take effect the next time a password is changed.

QRMTSIGN

Specifies how the system handles remote sign-on requests. A TELNET session is actually a remote sign-on request. This value determines several actions, as follows:

- ***FRCSIGNON**: All remote sign-on sessions are required to go through normal sign-on processing.
- ***SAMEPRF**: For 5250 display station pass-through or workstation function, when the source and target user profile names are the same, the sign-on may be bypassed for remote sign-on attempts. When using TELNET, the sign-on may be bypassed.
- ***VERIFY**: After verifying that the user has access to the system, the system allows the user to bypass the sign-on.
- ***REJECT**: Allows no remote sign-on for 5250 display station pass-through or workstation function. When QRMTSIGN is set to ***REJECT**, the user can still sign-on to the system by using TELNET. These sessions will go through normal processing. If you want to reject all TELNET requests to the system, end the TELNET servers.
- **'program library'**: The user can specify a program and library (or ***LIBL**) to decide which remote sessions are allowed and which user profiles can be automatically signed on from which locations. This option is only valid for passthrough.

This value also specifies a program name to run that determines which remote sessions are to be allowed.

The shipped value is ***FRCSIGNON**. If you want users to be able to use the bypass sign-on function of the 5250 emulator, change this value to ***VERIFY**.

QSECURITY

System security level. Changes that are made to this system value take effect at the next IPL.

- 20 means that the system requires a password to sign-on.
- 30 means that the system requires password security at sign-on and object security at each access. You must have authority to access all system resources.
- 40 means that the system requires password security at sign-on and object security at each access. Programs that try to access objects through unsupported interfaces fail.
- 50 means that the system requires password security at sign-on, and users must have authority to access objects and system resources. The security and integrity of the QTEMP library and user domain objects are enforced. Programs that try to access objects through interfaces that are not supported or that try to pass unsupported parameter values to supported interfaces will fail.

QSTRUPPGM

The program that runs when the controlling subsystem starts or when the system starts. This program performs set up functions such as starting subsystems.

QSYSLIBL

The system part of the library list. This part of the library list is searched before any other part. Some client functions use this list to search for objects.

Identifying Server Jobs on the iSeries server

You may find using an emulator or green screen interface makes it difficult to relate a job to a certain personal computer or an individual client function. Being able to identify a particular job is a prerequisite to investigating problems and determining performance implications. You can use the iSeries Navigator interface to identify your server jobs.

1. Double-click on the **iSeries Navigator** icon.
2. Open **Network** by clicking on the +.
3. Open **Servers** by clicking on the +.
4. Select the type of servers that you want to see jobs for (For example, TCP/IP or iSeries Access for Windows).
5. When the servers show in the right pane, right-click on the server that you want to see jobs for and click **Server Jobs**. Another window opens, showing the server jobs with the user, job type, job status, time entered system and date entered system for that server.

The following sections provides information on how to identify server jobs using the traditional green screen interface.

- Subsystems on the iSeries server
- iSeries Job Names
- Displaying Server Jobs
- Displaying the History Log
- Displaying server jobs for a user

iSeries Job Names: The job name that is used on the iSeries consists of three parts:

- The simple job name
- The user ID
- The job number (ascending order)

The server jobs follow several conventions:

- Job name
 - For non-prestarted jobs, the server job name is the name of the server program.
 - Prestarted jobs use the name that is defined in the prestart job entry.
 - Jobs that are started by the servers use the job description name or a given name if they are batch jobs (the file server does this).
- The user ID
 - Is always QUSER, regardless of whether prestart jobs are used.
 - The job log shows which users have used the job.
- Work management creates the job number.

Displaying Server Jobs: Two methods can be used to identify server jobs. The first method is to use the WRKACTJOB command. The second method is to display the history log to determine which job is being used by which client.

<h7>Displaying Active Jobs with WRKACTJOB

The WRKACTJOB command shows all active jobs, as well as the server daemons and the server mapper daemon.

The following figures show a sample status with the WRKACTJOB command. Only jobs related to the servers are shown in the figures. You must press **(F14)** to see the available prestart jobs.

The following types of jobs are shown in the figures:

- (1) - Server mapper daemon
- (2) - Server daemons
- (3) - Prestarted server jobs

```

-----+-----
                                Work with Active Jobs                                AS400597
                                01/12/95 10:25:40
CPU %:   3.1  Elapsed time: 21:38:40  Active jobs: 77

Type options, press Enter.
  2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
  8=Work with spooled files  13=Disconnect ...

Opt  Subsystem/Job  User      Type  CPU %  Function      Status
-----+-----
      .
___  QSYSWRK        QSYS      SBS    .0     DEQW
___  (1) QZSOSMAPD    QUSER     BCH    .0     SELW
      .
___  (2) QZSOSGND    QUSER     BCH    .0     SELW
___  QZSCSRVSD      QUSER     BCH    .0     SELW
___  QZRCSRVD       QUSER     BCH    .0     SELW
___  QZHQSRVD       QUSER     BCH    .0     SELW
___  QNPSERVD       QUSER     BCH    .0     SELW
      .
___  QUSRWRK        QSYS      SBS    .0     DEQW
___  (3) QZSOSIGN    QUSER     PJ     .0     PSRW
___  QZSCSRVS       QUSER     PJ     .0     PSRW
___  QZRCSRVS       QUSER     PJ     .0     PSRW
___  QZHQSSRV       QUSER     PJ     .0     PSRW
___  QNPSERVS       QUSER     PJ     .0     PSRW
___  QZDASOINIT     QUSER     PJ     .0     PSRW
      .
More...
-----+-----

```

```

-----+-----
                                Work with Active Jobs                                AS400597
                                01/12/95 10:25:40
CPU %:   3.1  Elapsed time: 21:38:40  Active jobs: 77

Type options, press Enter.
  2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
  8=Work with spooled files  13=Disconnect ...

Opt  Subsystem/Job  User      Type  CPU %  Function      Status
-----+-----
      .
___  QSERVER        QSYS      SBS    .0     DEQW
___  QSERVER        QPGMR     ASJ    .1     EVTW
      .
___  (2) QPWFSERVSD  QUSER     BCH    .0     SELW
___  QZDASRVSD      QUSER     BCH    .0     SELW
      .
___  (3) QPWFSERVSO  QUSER     PJ     .0     PSRW
___  QPWFSERVSO    QUSER     PJ     .0     PSRW
      .
More...
-----+-----

```

The following types of jobs are shown:

- ASJ** The autostart job for the subsystem
- PJ** The prestarted server jobs
- SBS** The subsystem monitor jobs
- BCH** The server daemon and the server mapper daemon jobs

Displaying the History Log: Each time a client user successfully connects to a server job, that job is swapped to run under the profile of that client user. To determine which job is associated with a particular client user, you can display the history log with the DSPLOG command. Look for the messages starting with:

- CPIAD0B (for signon server messages)
- CPIAD09 (for messages relating to all other servers)

Displaying server jobs for a user: To display the server jobs for a particular user,

1. Open **iSeries Navigator** (double-click on the icon).
2. Click on **Users and Groups**, then **All Users**.
3. Right-click on the user that you want to see server jobs for.
4. Select **User Objects**, then click on **Jobs**. You see a window displaying all the server jobs for that user.

You can also use the WRKOBJLCK command. Specify the user profile and *USRPRF.

Using EZ-Setup and iSeries Navigator with host servers

EZ-Setup and iSeries Navigator may connect to the sign-on, central, and remote command/distributed program call servers without a communication protocol running on the iSeries server. That is, EZ-Setup may connect before STRTCP has been run. The path used permits EZ-Setup to perform some initial iSeries setup before configuring or starting any communication protocols. This topic describes how to determine if the communication path used by EZ-Setup and Operations Console is active and how to restart it if necessary.

For information on configuring the connection that is used by EZ-Setup consult the EZ-Setup online help.

The communication path used by EZ-Setup requires three jobs, QNEOSOEM, to be running in the QSYSWRK subsystem. The QSYSWRK subsystem has an auto start job for this communication path. The auto start job, QNEOSOEM, submits two other jobs with the name of QNEOSOEM in the QSYSWRK subsystem. If one of the jobs is not active, start it by issuing the following command:

```
QSYS/SBMJOB CMD(QSYS/CALL PGM(QSYS/QNEOSOEM)) JOB(QNEOSOEM)
JOB(QSYS/QNEOJOB) JOB(QSYS/QSYSNOMAX) PRTDEV(*JOB) OUTQ(*JOB)
USER(*JOB) PRTTXT(*JOB) SYSLIBL(*SYSVAL) INLLIBL(*JOB)
LOGCLPGM(*YES) MSGQ(*NONE) SRTSEQ(*SYSVAL) LANGID(*SYSVAL)
CNTRYID(*SYSVAL) CCSID(*SYSVAL)
```

The command will start all three QNEOSOEM jobs if necessary.

Using server exit programs

Exit programs allow system administrators to control which activities a client user is allowed for each of the specific servers. All of the servers support user-written exit programs. This topic describes how the exit programs can be used, and how to configure them. It also provides sample programs that can help control access to server functions.

- Registering exit programs
- Writing exit programs
- Exit program parameters
- Example exit programs

Note: Read the Code example disclaimer for important legal information.

Registering Exit Programs

In order for the servers to know which exit program, if any, to call, you must register your exit program. You can register the exit program using the OS/400 registration facility.

Working with the Registration Facility

To register an exit program with the registration facility, use the Work with Registration Information (WRKREGINF) command.

```

+-----+
|                                     |
|                               Work with Registration Info (WRKREGINF)         |
|                                     |
| Type choices, press Enter.         |
|                                     |
| Exit point . . . . . *REGISTERED   |
| Exit point format . . . . . *ALL    | Name, generic*, *ALL |
| Output . . . . . *                  | *, *PRINT          |
|                                     |
+-----+

```

Press Enter to view the registered exit points.

```

+-----+
|                                     |
|                               Work with Registration Information             |
|                                     |
| Type options, press Enter.         |
| 5=Display exit point 8=Work with exit programs                             |
|                                     |
|                                     |
|                                     |
| Opt  Exit Point          Exit Point  Registered  Text                    |
|  _  QIBM_QGW_NJEOUBOUND  NJEO0100   *YES        Network Job Entry outb    |
|  8  QIBM_QHQ_DTAQ        DTAQ0100   *YES        Original Data Queue Se   |
|  _  QIBM_QLZP_LICENSE    LICM0100   *YES        Original License Mgmt    |
|  _  QIBM_QMF_MESSAGE     MESS0100   *YES        Original Message Serve   |
|  _  QIBM_QNPS_ENTRY      ENTR0100   *YES        Network Print Server -   |
|  _  QIBM_QNPS_SPLF       SPLF0100   *YES        Network Print Server -   |
|  _  QIBM_QNS_CRADDACT    ADDA0100   *YES        Add CRQ description ac   |
|  _  QIBM_QNS_CRCHGACT    CHGA0100   *YES        Change CRQ description   |
|  _  QIBM_QNS_CRDLTSBMCRQ DLTA0100   *YES        Delete submitted CRQ     |
|  _  QIBM_QNS_CRDSPACT    DSPA0100   *YES        Display CRQ description  |
|  _  QIBM_QNS_CREXCACT    EXCA0100   *YES        Run CRQ activity        |
|                                     |
| Command                             |
| ===>                                |
|                                     |
+-----+

```

Choose option 8 to work with the exit programs for the exit point defined for the server you would like to work with.

```

+-----+
|                                     |
|                               Work with Exit Programs                       |
|                                     |
| Exit point:  QIBM_QHQ_DTAQ          Format:  DTAQ0100                     |
|                                     |
| Type options, press Enter.         |
| 1=Add 4=Remove 5=Display 10=Replace |
|                                     |
|                                     |
|                                     |
| Opt  Exit Program      Exit Program  Library                    |
|  1_  _____          _____    _____                    |
|                                     |
| (No exit programs found)          |
|                                     |
+-----+

```

Use option 1 to add an exit program to an exit point.

Notes:

- If an exit program is already defined, you must remove it before you can change the name of the program.
- Even though the registration facility can support multiple user exits for a specific exit point and format name, the servers always retrieve exit program 1.
- You must end and restart the prestart jobs for the change to go into affect.

```
+-----+
|                                     Add Exit Program (ADDEXITPGM)
|
| Type choices, press Enter.
|
| Exit point . . . . . > QIBM_QHQ_DTAQ
| Exit point format . . . . . > DTAQ0100      Name
| Program number . . . . . > 1                1-2147483647, *LOW, *HIGH
| Program . . . . . MYPGM                      Name
|   Library . . . . . MYLIB Name, *CURLIB
| Text 'description' . . . . . *BLANK
|
+-----+
```

Enter your program name and library for the program at this exit point.

The same program is usable for multiple exit points. The program can use the data that is sent as input to determine how to handle different types of requests.

The following provides the exit point and format names for each of the specific OS/400 servers.

QIBM_QPWFS_FILE_SERV (File Server)

Format Name PWFS0100
Application Name *FILESRV

QIBM_QZDA_INIT (Database server initiation)

Format Name ZDAI0100
Application Name *SQL

QIBM_QZDA_NDB1 (Database server-native database requests)

Format Names ZDAD0100 ZDAD0200
Application Name *NDB

QIBM_QZDA_SQL1 (Database server SQL requests)

Format Names ZDAQ0100 ZDAQ0200
Application Name *SQLSRV

QIBM_QZDA_ROI1 (Database server retrieve object information requests)

Format Names ZDAR0100 ZDAR0200
Application Name *RTVOBJINF

QIBM_QZHQ_DATA_QUEUE (Data queue server)

Format Name ZHQ00100
Application Name *DATAQSRV

QIBM_QNPS_ENTRY (Network print server)

Format Name ENTR0100
Application Name QNPSERV

QIBM_QNPS_SPLF (Network print server)

Format Name SPLF0100
Application Name QNPSERV

QIBM_QZSC_LM (Central server license management requests)

Format Name ZSCL0100
Application Name *CNTRLSRV

QIBM_QZSC_NLS (Central server NLS requests)

Format Name ZSCN0100
Application Name *CNTRLSRV

QIBM_QZRC_RMT (Remote command and distributed program call server)

Format Name CZRC0100
Application Name *RMTSRV

QIBM_QZSO_SIGNONSRV (Signon server)

Format Name ZSOY0100
Application Name *Signon

Writing Exit Programs

When you specify an exit program the servers pass the following two parameters to the exit program before running your request:

- A 1-byte return code value
- A structure containing information about your request (This structure is different for each of the exit points.)

When you specify an exit program the servers pass the following two parameters to the exit program before running your request.

- A 1-byte return code value
- A structure containing information about your request (This structure is different for each of the exit points.)

These two parameters allow the exit program to determine whether your request is possible. If the exit program sets the return code to X'F1', the server allows the request. If the return code is set to X'F0' the server rejects the request. If values other than X'F1' or X'F0' are set, the results will vary depending upon which server is being accessed.

For multiple servers and exit points, the same program is usable. The program can determine which server is being called and which function is being used by looking at the data in the second parameter structure.

Parameter Formats for Exit Programs documents the structures of the second parameter that is sent to the exit programs. You can use this information to write your own exit programs.

Exit program parameters

These topics provide the data structure for the second parameter to the exit point formats for each of the OS/400 servers.

- File server
- Database server
- Data queue server
- Network print server
- Central server
- Remote command and distribute and distributed program call server
- Signon server

File Server: The file server has one exit point defined:

QIBM_QPWFS_FILE_SERV Format PWFS0100

The QIBM_QPWFS_FILE_SERV exit point is defined to run an exit program for the following types of file server requests:

- Change file attributes
- Create stream file or create directory
- Delete file or delete directory
- List file attributes
- Move
- Open stream file
- Rename
- Allocate conversation

Note:

For the file server, the exit program name is resolved when the QSERVER subsystem is activated. If you change the program name, you must end and restart the subsystem for the change to take effect.

Exit Point QIBM_QPWFS_FILE_SERV format PWFS0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the file server, the value is *FILESRV. |

| | | | | |
|----|----|-----------|--------------------|--|
| 20 | 14 | BINARY(4) | Requested function | <p>The function being performed:</p> <ul style="list-style-type: none"> • X'0000' - Change file attributes request • X'0001' - Create stream file or directory request • X'0002' - Delete file or delete directory request • X'0003' - List file attributes request • X'0004' - Move request • X'0005' - Open stream file request • X'0006' - Rename request • X'0007' - Allocate conversation request |
| 24 | 18 | CHAR(8) | Format name | <p>The user exit format name being used. For QIBM_QPWFS_FILE_SERV, the format name is PWFS0100.</p> |
| 32 | 20 | CHAR(4) | File access | <p>If the requested function has a value of '5' (open), this field contains the following structure:</p> <ul style="list-style-type: none"> • Read access, CHAR(1) X'F1' - Yes X'F0' - No • Write access, CHAR(1) X'F1' - Yes X'F0' - No • Read/Write access, CHAR(1) X'F1' - Yes X'F0' - No • Delete allowed, CHAR(1) X'F1' - Yes X'F0' - No |
| 36 | 24 | BINARY(4) | File name length | <p>The length of the file name (the next field). The length can be a maximum of 16MB.</p> |

| | | | | |
|----|----|---------|-----------|--|
| 40 | 28 | CHAR(*) | File name | The name of the file. The length of this field is specified by the File Name Length (the previous field). The file name is returned in the ISO/IEC 10646 (UCS—2 Level 1) character set, CCSID 61952. |
|----|----|---------|-----------|--|

Note:

- This format is defined by member EPWFSEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLLSRC in library QSYSINC.
- For more information about the ISO/IEC 10646 (UCS—2 Level 1) character set, see *Information Standard, ISO/IEC 10646—1: Information technology — Universal—Octet Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane*, reference number ISO/IEC 10646—1: 1993(E).

The APIs available to convert to and from UCS—2 Level 1 are iconv() and CDRCVRT.

Database server: The database server has four different exit points defined:

1. QIBM_QZDA_INIT
 - Called at server initiation
2. QIBM_QZDA_NDB1
 - Called for native database requests
3. QIBM_QZDA_SQL1
 - Called for SQL requests
4. QIBM_QZDA_SQL2
 - Called for SQL requests
5. QIBM_QZDA_ROI1
 - Called for retrieving object information requests and SQL catalog functions

The exit points for native database and retrieving object information have two formats defined depending on the type of function requested.

The QIBM_QZDA_INIT exit point is defined to run an exit program at server initiation. If a program is defined for this exit point, it is called each time the database server is initiated.

Exit Point QIBM_QZDA_INIT format ZDAI0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For this exit point, the value is *SQL. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZDA_INIT the format name is ZDAI0100. |

| | | | | |
|----|----|-----------|--------------------|--|
| 28 | 1C | BINARY(4) | Requested function | The function being performed |
| | | | | The only valid value for this exit point is 0. |

Note: This format is defined by member EZDAEP in files H, QRP GSRC, QRP GLE SRC, QL BLSRC and QCBLLESRC in library QSYSINC.

The QIBM_QZDA_NDB1 exit point is defined to run an exit program for native database requests for the database server. Two formats are defined for this exit point. Format ZDAD0100 is used for the following functions:

- Create source physical file
- Create database file, based on existing file
- Add, clear, delete database file member
- Override database file
- Delete database file override
- Delete file

Format ZDAD0200 is used when a request is received to add libraries to the library list.

Exit Point QIBM_QZDA_NDB1 format ZDAD0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For this exit point, the value is *NDB. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used |
| | | | | | For the following functions, the format name is ZDAD0100. |

| | | | | |
|-----|-----|-----------|-----------------------|--|
| 28 | 1C | BINARY(4) | Requested function | The function being performed This field contains one of the following: <ul style="list-style-type: none"> • X'1800' - Create source physical file • X'1801' - Create database file • X'1802' - Add database file member • X'1803' - Clear database file member • X'1804' - Delete database file member • X'1805' - Override database file • X'1806' - Delete database file override • X'1807' - Create save file • X'1808' - Clear save file • X'1809' - Delete file |
| 32 | 20 | CHAR(128) | File name | Name of the file used for the requested function |
| 160 | A0 | CHAR(10) | Library name | Name of the library that contains the file |
| 170 | AA | CHAR(10) | Member name | Name of the member to be added, cleared, or deleted |
| 180 | B4 | CHAR(10) | Authority | Authority to the created file |
| 190 | BE | CHAR(128) | Based on file name | Name of the file to use when creating a file based on an existing file |
| 318 | 13E | CHAR(10) | Based on library name | Name of the library containing the based on file |
| 328 | 148 | CHAR(10) | Override file name | Name of the file to be overridden |
| 338 | 152 | CHAR(10) | Override library name | Name of the library that contains the file to be overridden |
| 348 | 15C | CHAR(10) | Override member name | Name of the member to be overridden |

Note: This format is defined by member EZDAEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLLESRC in library QSYSINC.

Exit Point QIBM_QZDA_NDB1 format ZDAD0200

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|-----------|---------------------|--|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For this exit point, the value is *NDB. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For the add to library list function, the format name is ZDAD0200. |
| 28 | 1C | | BINARY(4) | Requested function | The function being performed |
| | | | | | X'180C' - Add library list |
| 32 | 20 | | BINARY(4) | Number of libraries | The number of libraries (the next field) |
| 36 | 24 | | CHAR(10) | Library name | The library names for each library |

Note: This format is defined by member EZDAEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLLESRC in library QSYSINC.

The QIBM_QZDA_SQL1 exit point is defined to run an exit point for certain SQL requests that are received for the database server. Only one format is defined for this exit point. The following are the functions that cause the exit program to be called:

- Prepare
- Open
- Execute
- Connect
- Create package
- Clear package
- Delete package
- Stream fetch
- Execute immediate
- Prepare and describe
- Prepare and execute or prepare and open
- Open and fetch
- Execute or open

Exit Point QIBM_QZDA_SQL1 format ZDAQ0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |

| | | | | |
|----|----|-----------|-------------------------------|---|
| 10 | A | CHAR(10) | Server identifier | For this exit point, the value is *SQLSRV. |
| 20 | 14 | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZDA_SQL1, the format name is ZDAQ0100. |
| 28 | 1C | BINARY(4) | Requested function | The function being performed This field contains one of the following: <ul style="list-style-type: none"> • X'1800' - Prepare • X'1803' - Prepare and describe • X'1804' - Open/Describe • X'1805' - Execute • X'1806' - Execute immediate • X'1809' - Connect • X'180C' - Stream fetch • X'180D' - Prepare and execute • X'180E' - Open and fetch • X'180F' - Create package • X'1810' - Clear package • X'1811' - Delete package • X'1812' - Execute or open |
| 32 | 20 | CHAR(18) | Statement name | Name of the statement used for the prepare or execute functions |
| 50 | 32 | CHAR(18) | Cursor name | Name of the cursor used for the open function |
| 68 | 44 | CHAR(2) | Prepare option | Option used for the prepare function |
| 70 | 46 | CHAR(2) | Open attributes | Option used for the open function |
| 72 | 48 | CHAR(10) | Extended dynamic package name | Name of the extended dynamic SQL package |
| 82 | 52 | CHAR(10) | Package library name | Name of the library for extended dynamic SQL package. |

| | | | | |
|----|----|-----------|---|--|
| 92 | 5C | BINARY(2) | DRDA indicator | <ul style="list-style-type: none"> • 0 - Connected to local RDB • 1 - Connected to remote RDB |
| 94 | 5E | CHAR(1) | Commitment control level | <ul style="list-style-type: none"> • 'A' - Commit *ALL • 'C' - Commit *CHANGE • 'N' - Commit *NONE • 'S' - Commit *CS (cursor stability) |
| 95 | 5F | CHAR(512) | First 512 bytes of the SQL statement text | First 512 bytes of the SQL statement |

Note: This format is defined by member EZDAEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLLSRC in library QSYSINC.

The QIBM_QZDA_SQL2 exit point is defined to run an exit point for certain SQL requests that are received for the database server. The QIBM_QZDA_SQL2 exit point takes precedence over the QIBM_QZDA_SQL1 exit point. If a program is registered for the QIBM_QZDA_SQL2 exit point, it will be called and a program for the QIBM_QZDA_SQL1 exit point will not be called. The following are the functions that cause the exit program to be called:

- Prepare
- Open
- Execute
- Connect
- Create package
- Clear package
- Delete package
- Stream fetch
- Execute immediate
- Prepare and describe
- Prepare and execute or prepare and open
- Open and fetch
- Execute or open

Table A-6. Exit Point QIBM_QZDA_SQL2 format ZDAQ0200

| | | | | |
|----|----|----------|-------------------|--|
| 0 | 0 | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | CHAR(10) | Server identifier | For this exit point, the value is *SQLSRV. |
| 20 | 14 | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZDA_SQL1, the format name is ZDAQ0100. |

| | | | | |
|-----|----|-----------|-------------------------------|---|
| 28 | 1C | BINARY(4) | Requested function | The function being performed This field contains one of the following: <ul style="list-style-type: none"> • X'1800' - Prepare • X'1803' - Prepare and describe • X'1804' - Open/Describe • X'1805' - Execute • X'1806' - Execute immediate • X'1809' - Connect • X'180C' - Stream fetch • X'180D' - Prepare and execute • X'180E' - Open and fetch • X'180F' - Create package • X'1810' - Clear package • X'1811' - Delete package • X'1812' - Execute or open |
| 32 | 20 | CHAR(18) | Statement name | Name of the statement used for the prepare or execute functions |
| 50 | 32 | CHAR(18) | Cursor name | Name of the cursor used for the open function |
| 68 | 44 | CHAR(2) | Prepare option | Option used for the prepare function |
| 70 | 46 | CHAR(2) | Open attributes | Option used for the open function |
| 72 | 48 | CHAR(10) | Extended dynamic package name | Name of the extended dynamic SQL package |
| 82 | 52 | CHAR(10) | Package library name | Name of the library for extended dynamic SQL package. |
| 92 | 5C | BINARY(2) | DRDA indicator | <ul style="list-style-type: none"> • 0 - Connected to local RDB • 1 - Connected to remote RDB |
| 94 | 5E | CHAR(1) | Commitment control level | <ul style="list-style-type: none"> • 'A' - Commit *ALL • 'C' - Commit *CHANGE • 'N' - Commit *NONE • 'S' - Commit *CS (cursor stability) |
| 95 | 5F | CHAR(10) | Default SQL collection | Name of the default SQL collection used by the iSeries Database Server |
| 105 | 69 | CHAR(129) | Reserved | Reserved for future parameters |
| 234 | EA | BINARY(4) | SQL statement text length | Length of SQL statement text in the field that follows. The length can be a maximum of 32K. |
| 238 | EE | CHAR(*) | SQL statement text | Entire SQL statement |

Note: This format is defined by member EZDAEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLESRC in library QSYSINC.

The QIBM_QZDA_ROI1 exit point is defined to run an exit program for the requests that retrieve information about certain objects for the database server. It is also used for SQL catalog functions.

This exit point has two formats defined. These formats are described below.

Format ZDAR0100 is used for requests to retrieve information for the following objects:

- Library (or collection)
- File (or table)

- Field (or column)
- Index
- Relational database (or RDB)
- SQL package
- SQL package statement
- File member
- Record format
- Special columns

Format ZDAR0200 is used for requests to retrieve information for the following objects:

- Foreign keys
- Primary keys

Exit Point QIBM_QZDA_ROI1 format ZDAR0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the database server, the value is *RTVOBJINF. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For the following functions, the format name is ZDAR0100. |

| | | | | |
|-----|----|-----------|----------------------------|---|
| 28 | 1C | BINARY(4) | Requested function | <p>The function being performed</p> <p>This field contains one of the following:</p> <ul style="list-style-type: none"> • X'1800' - Retrieve library information • X'1801' - Retrieve relational database information • X'1802' - Retrieve SQL package information • X'1803' - Retrieve SQL package statement • X'1804' - Retrieve file information • X'1805' - Retrieve file member information • X'1806' - Retrieve record format information • X'1807' - Retrieve field information • X'1808' - Retrieve index information • X'180B' - Retrieve special column information |
| 32 | 20 | CHAR(20) | Library name | <p>The library or search pattern used when retrieving information about libraries, packages, package statements, files, members, record formats, fields, indexes, and special columns</p> |
| 52 | 34 | CHAR(36) | Relational database name | <p>The relational database name or search pattern used to retrieve RDB information</p> |
| 88 | 58 | CHAR(20) | Package name | <p>The package name or search pattern used to retrieve package or package statement information</p> |
| 108 | 6C | CHAR(256) | File name (SQL alias name) | <p>The file name or search pattern used to retrieve file, member, record format, field, index, or special column information</p> |

| | | | | |
|-----|-----|----------|-------------|--|
| 364 | 16C | CHAR(20) | Member name | The member name or search pattern used to retrieve file member information |
| 384 | 180 | CHAR(20) | Format name | The format name or search pattern used to retrieve record format information |

Note: This format is defined by member EZDAEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLLSRC in library QSYSINC.

Exit Point QIBM_QZDA_ROI1 format ZDAR0200

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|-----------|-------------------------------------|--|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the database server, the value is *RTVOBJINF. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For the following functions, the format name is ZDAR0200. |
| 28 | 1C | | BINARY(4) | Requested function | The function being performed This field contains one of the following: <ul style="list-style-type: none"> • X'1809' - Retrieve foreign key information • X'180A' - Retrieve primary key information |
| 32 | 20 | | CHAR(10) | Primary key table library name | The name of the library that contains the primary key table used when retrieving primary and foreign key information |
| 42 | 2A | | CHAR(128) | Primary key table name (alias name) | The name of the table that contains the primary key used when retrieving primary or foreign key information |
| 170 | AA | | CHAR(10) | Foreign key table library name | The name of the library that contains the foreign key table used when retrieving foreign key information |

| | | | | |
|-----|----|-----------|-------------------------------------|--|
| 180 | 64 | CHAR(128) | Foreign key table name (alias name) | The name of the table that contains the foreign key used when retrieving foreign key information |
|-----|----|-----------|-------------------------------------|--|

Note: This format is defined by member EZDAEP in files H, QRPGRSRC, QRPGLSRC, QLBSRC and QCBLESRC in library QSYSINC.

Data Queue Server: The data queue server has one exit point defined:

QIBM_QZHQ_DATA_QUEUE format ZHQ00100

The exit point QIBM_QZHQ_DATA_QUEUE is defined to run an exit point program when the following data queue server requests are received:

- Query
- Receive
- Create
- Delete
- Send
- Clear
- Cancel
- Peek

Exit Point QIBM_QZHQ_DATA_QUEUE format ZHQ00100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the data queue, server the value is *DATAQSRV. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZHQ_DATA_QUEUE the format name is ZHQ00100. |

| | | | | |
|----|----|-----------|----------------------|---|
| 28 | 1C | BINARY(4) | Requested function | The function being performed <ul style="list-style-type: none"> • X'0001' - Query the attributes of a data queue • X'0002' - Receive a message from a data queue • X'0003' - Create a data queue • X'0004' - Delete a data queue • X'0005' - Send a message to a data queue • X'0006' - Clear messages from a data queue • X'0007' - Cancel a pending receive request • X'0012' - Receive a message from a data queue without deleting it |
| 32 | 20 | CHAR(10) | Object name | Data queue name |
| 42 | 2A | CHAR(10) | Library name | Data queue library |
| 52 | 34 | CHAR(2) | Relational operation | Relational operator for receive-by-key operation on the request <ul style="list-style-type: none"> X'0000' - No operator 'EQ' - Equal 'NE' - Not equal 'GE' - Greater or equal 'GT' - Greater than 'LE' - Less or equal 'LT' - Less than |
| 54 | 36 | BINARY(4) | Key length | Key length specified on the request |
| 58 | 3A | CHAR(256) | Key value | Key value specified on the request |

Note: This format is defined by member EZHQEP in files H, QRPGRSRC, QRPGLSRC, QLBSRC and QCBLESRC in library QSYSINC.

Central Server: The central server has three exit points defined:

1. QIBM_QZSC_LM format ZSCL0100
 - Called for license management requests
2. QIBM_QZSC_SM format ZSCS0100
 - Called for system management requests
3. QIBM_QZSC_NLS format ZSCN0100
 - Called for conversion table requests

The QIBM_QZSC_LM exit point is defined to run an exit program for all license management requests received by the central server.

Exit Program QIBM_QZSC_LM format ZSCL0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|-----------|---------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the central server, the value is *CNTRLSRV. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZSC_LM, the format name is ZSCL0100. |
| 28 | 1C | | BINARY(4) | Requested function | The function being performed This field contains one of the following: <ul style="list-style-type: none"> • X'1001' - Request license • X'1002' - Release license • X'1003' - Retrieve license information |
| 32 | 20 | | CHAR(255) | Unique client name | The unique client name is used to identify a specific workstation across a network. The use of a licensed product is assigned to a workstation identified by the unique client name. |
| 287 | 11F | | CHAR(8) | License user handle | License user handle is used to ensure that the license requester and license releaser are the same. This value must be the same as when the license was requested. |

| | | | | |
|-----|-----|-----------|------------------------|--|
| 295 | 127 | CHAR(7) | Product identification | The identification of the product whose licensed use is requested |
| 302 | 12E | CHAR(4) | Feature identification | The feature of the product |
| 306 | 132 | CHAR(6) | Release identification | The version, release, and modification level of the product or feature |
| 312 | 138 | BINARY(2) | Type of information | The type of information to be retrieved. The type of information field is only valid for the retrieve license information function This field contains one of the following: <ul style="list-style-type: none"> • X'0000' - Basic license information • X'0001' - Detailed license information |

Note: This format is defined by member EZSCEP in files H, QRPGRSRC, QRPGLSRC, QLBLSRC and QCBLLSRC in library QSYSINC.

The QIBM_QZSC_SM exit point is defined to run an exit program for all client management requests received by the central server.

Exit Program QIBM_QZSC_SM format ZSCS0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|-----------|--------------------|--|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the central server, the value is *CNTRLSRV. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZSC_SM the format name is ZSCS0100. |
| 28 | 1C | | BINARY(4) | Requested function | The function being performed This field contains one of the following: <ul style="list-style-type: none"> • X'1101' - Set client active • X'1102' - Set client inactive |

| | | | | |
|-----|-----|-----------|--------------------|---|
| 32 | 20 | CHAR(255) | Unique client name | The client workstation name that is assigned to the licensed product |
| 287 | 11F | CHAR(255) | Community name | The community name SNMP configuration field is used for authentication. |
| 542 | 21E | CHAR(1) | Node type | The type of connection <ul style="list-style-type: none"> • 3 - Internet |
| 543 | 21F | CHAR(255) | Node name | The name of the node For node type 3, the node name will be an Internet address. |

Note: This format is defined by member EZSCEP in files H, QRPGRSRC, QRPGLSRC, QLBSRC and QCBLESRC in library QSYSINC.

The QIBM_QZSC-NLS exit point is defined to run an exit program when the central server receives a request to retrieve a conversion map.

Exit Program QIBM_QZSC-NLS format ZSCN0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|-----------|---|--|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the central server, the value is *CNTRLSRV. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZSC-NLS, the format name is ZSCN0100. |
| 28 | 1C | | BINARY(4) | Requested function | The function being performed <ul style="list-style-type: none"> • X'1201' - Retrieve conversion map |
| 32 | 20 | | BINARY(4) | From coded character set identifier (CCSID) | CCSID for existing data |
| 36 | 24 | | BINARY(4) | To coded character set identifier (CCSID) | CCSID into which the data will be converted |
| 40 | 28 | | BINARY(2) | Type of conversion | Requested mapping type: <ul style="list-style-type: none"> • X'0001' - Round trip • X'0002' - Substitution mapping • X'0003' - Best-fit mapping |

Note: This format is defined by member EZSCEP in files H, QRPGRSRC, QRPGLSRC, QLBSRC and QCBLESRC in library QSYSINC.

Remote Command and Distributed Program Call Server: The remote command/distributed program call server has one exit point defined:

QIBM_QZRC_RMT format CZRC0100

The QIBM_QZRC_RMT exit point is defined to call a program for either remote command or distributed program call requests.

The format of the parameter fields differ according to the type of request.

Remote Command requests for Exit Point QIBM_QZRC_RMT format CZRC0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|-------------|--------------------------|---|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the remote command server, the value is *RMTSRV. |
| 20 | 14 | | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZRC_RMT, the format name is CZRC0100. |
| 28 | 1C | | BINARY(4) | Requested function | The function being performed |
| | | | | | X'1002' - Remote command |
| 32 | 20 | | CHAR(10) | Reserved | Not used for remote command requests |
| 42 | 2A | | CHAR(10) | Reserved | Not used for remote command requests |
| 52 | 34 | | BINARY(4) | Length of the next field | The length of the following command string |
| 56 | 38 | | CHAR (6000) | Command string | Command string for remote command requests |

Distributed Program Call requests for Exit Point QIBM_QZRC_RMT format CZRC0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|----------|-------------------|--|
| 0 | 0 | | CHAR(10) | User profile name | The name of the user profile that is calling the server |
| 10 | A | | CHAR(10) | Server identifier | For the distributed program call server, the value is *RMTSRV. |

| | | | | |
|----|----|-----------|----------------------|--|
| 20 | 14 | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZRC_RMT, the format name is CZRC0100. |
| 28 | 1C | BINARY(4) | Requested function | The function being performed |
| | | | | X'1003' - Distributed program call |
| 32 | 20 | CHAR(10) | Program name | Name of the program being called |
| 42 | 2A | CHAR(10) | Library name | Library of the specified program |
| 52 | 34 | BINARY(4) | Number of parameters | The total number of parameters for the program call. This does not always indicate the number of parameters that follow. |

| | |
|-----------------------|---|
| Parameter information | <p>Information about the parameters being passed to the specified program. All parameter strings have the following format regardless of the parameter usage type. The last field in the structure is specified for input/output parameter usage types.</p> <ul style="list-style-type: none"> • BINARY(4) - Length of parameter information for this parameter • BINARY(4) - Maximum length of parameter • BINARY(2) - Parameter usage type <ul style="list-style-type: none"> - 1 - Input - 2 - Output - 3 - Input / output • CHAR(*) - Parameter string <p>The maximum length of the parameter information is 6000 bytes. If the parameter information exceeds 6000 bytes, it is truncated.</p> |
|-----------------------|---|

Signon Server: The signon server, has one exit point defined:

QIBM_QZSO_SIGNONSRV format ZSOY0100

The exit point QIBM_QZSO_SIGNONSRV is defined to run an exit point program when the following signon server requests are received:

- Retrieve sign-on information
- Change password
- Generate authentication token

Exit Point QIBM_QZSO_SIGNONSRV format ZSOY0100

| Dec | Offset | Hex | Type | Field | Description |
|-----|--------|-----|------|-------|-------------|
|-----|--------|-----|------|-------|-------------|

| | | | | |
|----|----|-----------|--------------------|--|
| 0 | 0 | CHAR(10) | User profile name | The name of the user profile associated with the request |
| 10 | A | CHAR(10) | Server identifier | For the signon server, the value is *SIGNON. |
| 20 | 14 | CHAR(8) | Format name | The user exit format name being used. For QIBM_QZSO_SIGNONSRV, the format name is ZSOY0100. |
| 28 | 1C | BINARY(4) | Requested function | The function being performed <ul style="list-style-type: none"> • X'7004' - Retrieve sign-on information • X'7005' - Change password • X'7007' - Generate authentication token |

Examples: Exit Programs

The sample exit programs in this section do not show all possible programming considerations or techniques, but you can review the examples before you begin your own design and coding.

Code example disclaimer

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

All sample code is provided by IBM for illustrative purposes only. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

All programs contained herein are provided to you "AS IS" without any warranties of any kind. The implied warranties of non-infringement, merchantability and fitness for a particular purpose are expressly disclaimed.

- Examples: Creating Exit Programs with RPG
- Examples: Creating Exit Programs with Control Language

e disclaimer: `"../synchron.js">` Examples: Creating Exit Programs with RPG

The following example illustrates how to set up a user exit program with RPG*.

Note: Read the Code example disclaimer for important legal information.

```

**
** OS/400 SERVERS - SAMPLE USER EXIT PROGRAM
**
** THE FOLLOWING RPG PROGRAM UNCONDITIONALLY
** ACCEPTS ALL REQUESTS. IT CAN BE USED AS A SHELL
** FOR SPECIFIC APPLICATIONS. NOTE: REMOVE THE
** SUBROUTINES AND CASE STATEMENT ENTRIES FOR THE SERVERS
** THAT DO NOT REQUIRE
** SPECIFIC EXIT PROGRAM HANDLING FOR BETTER PERFORMANCE.
**
E*
```

```

E* NECESSARY ARRAY DEFINITIONS FOR TRANSFER FUNCTION
E* AND REMOTE SQL
E*
E          TFREQ    4096  1
E          RSREQ    4107  1
I*
I*
IPCSDTA    DS
I          1  10  USERID
I          11 20  APPLID
I*
I* SPECIFIC PARAMETERS FOR VIRTUAL PRINTER
I*
I          21 30  VPFUNC
I          31 40  VPOBJ
I          41 50  VPLIB
I          71 750VPIFN
I          76 85  VPOUTQ
I          86 95  VPQLIB
I*
I* SPECIFIC PARAMETERS FOR MESSAGING FUNCTION
I          21 30  MFFUNC
I*
I* SPECIFIC PARAMETERS FOR TRANSFER FUNCTION
I*
I          21 30  TFFUNC
I          31 40  TFOBJ
I          41 50  TFLIB
I          51 60  TFMBR
I          61 70  TFFMT
I          71 750TFLEN
I          764171 TFREQ
I*
I* SPECIFIC PARAMETERS FOR FILE SERVER
I*
I* NOTE: FSNAME MAY BE UP TO 16MB.
I* FSNLEN WILL CONTAIN THE ACTUAL SIZE OF FSNAME.
I*
I          B 21 240FSFID
I          25 32  FSFMT
I          33 33  FSREAD
I          34 34  FSWRIT
I          35 35  FSRDWR
I          36 36  FSDLT
I          B 37 400FSNLEN
I          41 296 FSNAME
I*
I* SPECIFIC PARAMETERS FOR DATA QUEUES
I*
I          21 30  DQFUNC
I          31 40  DQQ
I          41 50  DQLIB
I          70 750DQLEN
I          76 77  DQROP
I          78 820DQKLEN
I          83 338 DQKEY
I*
I* SPECIFIC PARAMETERS FOR REMOTE SQL
I*
I          21 30  RSFUNC
I          31 40  RSOBJ
I          41 50  RSLIB
I          51 51  RSCMT
I          52 52  RSMODE
I          53 53  RSCID
I          54 71  RSSTN
I          72 75  RSRSV

```

```

I                               764182 RSREQ
I*
I* SPECIFIC PARAMETERS FOR NETWORK PRINT SERVER
I*
I                               21 28 NPFT
I                               B 29 320NPFID
I* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT SPLF0100
I                               33 42 NPJOB#
I                               43 52 NPUSR#
I                               53 58 NPJOB#
I                               59 68 NPFILE
I                               B 69 720NPFIL#
I                               B 73 760NPLEN
I                               77 332 NPDATA
I*
I* Data Queue server:
I*
I* QIBM_QZHQ_DATA_QUEUE format ZHQ00100
I*
I                               21 28 DQOFMT
I                               B 29 320DQOFID
I                               33 42 DQO0BJ
I                               43 52 DQOLIB
I                               53 54 DQOROP
I                               B 55 580DQOLEN
I                               59 314 DQOKEY
I*
I* Specific PARAMETERS FOR CENTRAL SERVER
I*
I                               21 28 CSFMT
I                               B 29 320CSFID
I* Central server:
I*
I* QIBM_QZSC_LM format ZSCL0100 for license management calls
I*
I*
I                               33 287 CSLCNM
I                               288 295 CSLUSR
I                               296 302 CSLPID
I                               303 306 CSLFID
I                               307 312 CSLRID
I                               B 313 3140CSLTYP
I*
I* Central server:
I*
I* QIBM_QZSC_LM format ZSCS0100 for system management calls
I*
I*
I                               33 287 CSSCNM
I                               288 542 CSSCMY
I                               543 543 CSSNDE
I                               544 798 CSSNNM
I*
I* Central server:
I*
I* QIBM_QZSC_LM format ZSCN0100 for retrieve conversion map calls
I*
I*
I                               21 30 CSNXFM
I                               29 320CSNFNC
I                               B 33 360CSNFRM
I                               B 37 400CSNTO
I                               B 41 420CSNCNT
I*
I* SPECIFIC PARAMETERS FOR DATABASE SERVER
I*

```



```

I          21 28 DBFMT
I          B 29 320DBFID
I*
I* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAD0100
I          33 160 DBDFIL
I          161 170 DBDLIB
I          171 180 DBDMBR
I          181 190 DBDAUT
I          191 318 DBDBFL
I          319 328 DBDBLB
I          329 338 DBDOFL
I          339 348 DBDOLB
I          349 358 DBDOMB
I*
I* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAD0200
I          B 33 360DBNUM
I          37 46 DBLIB2
I*
I* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAQ0100
I          33 50 DBSTMT
I          51 68 DBCRSR
I          69 70 DBOPI
I          71 72 DBATTR
I          73 82 DBPKG
I          83 92 DBPLIB
I          B 93 940DBDRDA
I          95 95 DBCMT
I          96 351 DBTEXT
I* THE FOLLOWING PARAMETERS REPLACE DBTEXT FOR FORMAT ZDAQ0200
I          96 105 DBSQCL
I          B 133 1360DBSQLN
I          137 392 DBSQTX
I* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAR0100
I          33 52 DBLIBR
I          53 88 DBRDBN
I          89 108 DBPKGR
I          109 364 DBFILR
I          365 384 DBMBRR
I          385 404 DBFFT
I* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAR0200
I          33 42 DBRPLB
I          43 170 DBRPTB
I          171 180 DBRFLB
I          181 308 DBRFTB
I*
I* Remote Command/Distributed Program Call server:
I*
I* QIBM_QZRC_RMT format CZRC0100
I*   RCPGM AND RCLIB ARE NOT USED FOR REMOTE COMMAND CALLS
I*
I          21 28 RCFMT
I          B 29 320RCFID
I          33 42 RCPGM
I          43 52 RCLIB
I          B 53 560RCNUM
I          57 312 RCDATA
I*
I* signon server:
I*
I* QIBM_QZSO_sign-onSRV format ZSOY0100 for TCP/IP signon server
I*
I          21 28 SOXFMT
I          B 29 320SOFID
I*
I*****
I*
I          '*VPRT'          C          #VPRT

```

```

I          '*TRFCL '          C          #TRFCL
I          '*FILESRV '        C          #FILE
I          '*MSGFCL '          C          #MSGF
I          '*DQSRV '           C          #DQSRV
I          '*RQSRV '           C          #RQSRV
I          '*SQL '             C          #SQL
I          '*NDB '             C          #NDBSV
I          '*SQLSRV '          C          #SQLSV
I          '*RTVOBJINF'        C          #RTVOB
I          '*DATAQSRV '        C          #DATAQ
I          '*QNPSERV '         C          #QNPSV
I          '*CNTRLSRV '        C          #CNTRL
I          '*RMTSRV '          C          #RMTSV
I          '*sign-on '         C          #SIGN
I*
C*
C* EXIT PROGRAM CALL PARAMETERS
C*
C          *ENTRY    PLIST
C                   PARM          RTNCD  1
C                   PARM          PCSDTA
C*
C* INITIALIZE RETURN VALUE TO ACCEPT REQUEST
C*
C                   MOVE '1'      RTNCD
C*
C* COMMON PROCESSING
C*
C*           COMMON LOGIC GOES HERE
C*
C* PROCESS BASED ON SERVER ID
C*
C          APPLID    CASEQ#VPRT    VPRT
C          APPLID    CASEQ#TRFCL   TFR
C          APPLID    CASEQ#FILE    FILE
C          APPLID    CASEQ#MSGF    MSG
C          APPLID    CASEQ#DQSRV   DATAQ
C          APPLID    CASEQ#RQSRV   RSQ
C          APPLID    CASEQ#SQL     SQLINT
C          APPLID    CASEQ#NDBSV   NDB
C          APPLID    CASEQ#SQLSV   SQLSRV
C          APPLID    CASEQ#RTVOB   RTVOBJ
C          APPLID    CASEQ#DATAQ   ODATAQ
C          APPLID    CASEQ#QNPSV   NETPRT
C          APPLID    CASEQ#CNTRL   CENTRL
C          APPLID    CASEQ#RMTSV   RMTCMD
C          APPLID    CASEQ#SIGN    sign-on
C          END
C                   SETON          LR
C                   RETRN
C*
C* SUBROUTINES
C*
C* VIRTUAL PRINT
C*
C          VPRT      BEGSR
C*           SPECIFIC LOGIC GOES HERE
C                   ENDSR
C*
C* TRANSFER FUNCTION
C*
C* THE FOLLOWING IS AN EXAMPLE OF SPECIFIC PROCESSING
C* THAT THE EXIT PROGRAM COULD DO FOR TRANSFER FUNCTION.
C*

```

```

C* IN THIS CASE, USERS ARE NOT ALLOWED TO SELECT
C* DATA FROM ANY FILES THAT ARE IN LIBRARY QIWS.
C*
C      TFR      BEGSR
C      TFFUNC   IFEQ 'SELECT'
C      TFLIB    ANDEQ'QIWS'
C              MOVE '0'      RTNCD
C              END
C              ENDSR
C*
C*
C* FILE SERVER
C*
C      FILE      BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* MESSAGING FUNCTION
C*
C      MSG      BEGSR
C*             SPECIFIC LOGIC GOFS HERE
C              ENDSR
C* DATA QUEUES
C*
C      DATAQ   BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* REMOTE SQL
C*
C      RSQL     BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* SERVERS
C*
C* DATABASE INIT
C*
C      SQLINT   BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* DATABASE NDB (NATIVE DATABASE)
C*
C      NDB      BEGSR
C*             SFECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* DATABASE SQL
C*
C      SQLSRV   BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* DATABASE RETRIEVE OBJECT INFORMATION
C*
C      RTVOBJ   BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*
C* DATA QUEUE SERVER
C*
C      ODATAQ   BEGSR
C*             SPECIFIC LOGIC GOES HERE
C              ENDSR
C*

```

```

C* NETWORK PRINT
C*
C          NETPRT    BEGSR
C*          SPECIFIC LOGIC GOES HERE
C          ENDSR
C*
C* CENTRAL SERVER
C*
C*
C* THE FOLLOWING IS AN EXAMPLE OF SPECIFIC PROCESSING
C* THAT THE EXIT PROGRAM COULD DO FOR LICENSE MANAGEMENT.
C*
C* IN THIS CASE, THE USER "USERALL" WILL NOT BE ALLOWED
C* TO EXECUTE ANY FUNCTIONS THAT ARE PROVIDED BY THE
C* CENTRAL SERVER FOR WHICH THIS PROGRAM IS A REGISTERED
C* EXIT PROGRAM - LICENSE INFORMATION, SYSTEM MANAGEMENT
C* OR RETRIVE A CONVERSION MAP.
C*
C          CENTRL    BEGSR
C          USERID    IFEQ 'USERALL'
C                   MOVE '0'          RTNCD
C                   ENDFIF
C*          SPECIFIC LOGIC GOES HERE
C          ENDSR
C*
C* REMOTE COMMAND/DISTRIBUTED PROGRAM CALL
C*
C* IN THIS CASE, THE USER "USERALL" WILL NOT BE ALLOWED
C* TO EXECUTE ANY REMOTE COMMANDS OR REMOTE PROGRAM CALLS
C*
C          RMTCMD    BEGSR
C          USERID    IFEQ 'USERALL'
C                   MOVE '0'          RTNCD
C                   ENDFIF
C          ENDSR
C*
C* sign-on SERVER
C*
C          sign-on    BEGSR
C*          SPECIFIC LOGIC GOES HERE
C          ENDSR

```

Examples: Creating Exit Programs with Control Language: The following example illustrates how to set up a user exit program control language (CL).

Note: Read the Code example disclaimer for important legal information.

```

/*****/
/*                                           */
/* iSeries SERVERS- SAMPLE USER EXIT PROGRAM */
/*                                           */
/* THE FOLLOWING CONTROL LANGUAGE PROGRAM UNCONDITIONALLY */
/* ACCEPTS ALL REQUESTS. IT CAN BE USED AS A SHELL FOR DEVELOPING */
/* EXIT PROGRAMS TAILORED FOR YOUR OPERATING ENVIRONMENT. */
/*                                           */
/*                                           */
/*****/
PGM PARM(&STATUS &REQUEST)

/* * * * * * * * * * * * * * * * * * */
/*                                           */
/* PROGRAM CALL PARAMETER DECLARATIONS */
/*                                           */
/* * * * * * * * * * * * * * * * * * */

DCL VAR(&STATUS) TYPE(*CHAR) LEN(1) /* Accept/Reject indicator */
/* */

```

```

/* Note: Request is declared as *CHAR LEN(2000) because that is */
/* the limit in CL. The actual length of REQUEST is 4171. */
/* */
DCL VAR(&REQUEST) TYPE(*CHAR) LEN(2000) /* Parameter structure */
/*****/
/* */
/* PARAMETER DECLARES */
/* */
/*****/

/* COMMON DECLARES */
DCL VAR(&USER) TYPE(*CHAR) LEN(10)
/* User ID */
DCL VAR(&APPLIC) TYPE(*CHAR) LEN(10)
/* Server ID */
DCL VAR(&FUNCTN) TYPE(*CHAR) LEN(10) /* Function being performed */

/* VIRTUAL PRINT DECLARES */
DCL VAR(&VPOBJ) TYPE(*CHAR) LEN(10) /* Object name */
DCL VAR(&VPLIB) TYPE(*CHAR) LEN(10) /* Object library name */
DCL VAR(&VPLEN) TYPE(*DEC) LEN(5 0) /* Length of following fields*/
DCL VAR(&VPOUTQ) TYPE(*CHAR) LEN(10) /* Output queue name */
DCL VAR(&VPQLIB) TYPE(*CHAR) LEN(10) /* Output queue library name */

/* TRANSFER FUNCTION DECLARES */
DCL VAR(&TFOBJ) TYPE(*CHAR) LEN(10) /* Object name */
DCL VAR(&TFLIB) TYPE(*CHAR) LEN(10) /* Object library name */
DCL VAR(&TFMBR) TYPE(*CHAR) LEN(10) /* Member name */
DCL VAR(&TFMT) TYPE(*CHAR) LEN(10) /* Record format name */
DCL VAR(&TFLEN) TYPE(*DEC) LEN(5 0) /* Length of request */
DCL VAR(&TFREQ) TYPE(*CHAR) LEN(1925) /*Transfer request statement*/

/* FILE SERVER DECLARES */
DCL VAR(&FSFID) TYPE(*CHAR) LEN(4) /* Function identifier */
DCL VAR(&FSFMT) TYPE(*CHAR) LEN(8) /* Parameter format */
DCL VAR(&FSREAD) TYPE(*CHAR) LEN(1) /* Open for read */
DCL VAR(&FSWRITE) TYPE(*CHAR) LEN(1) /* Open for write */
DCL VAR(&FSRDWRT) TYPE(*CHAR) LEN(1) /* Open for read/write */
DCL VAR(&FSDLT) TYPE(*CHAR) LEN(1) /* Open for delete */
DCL VAR(&FSLEN) TYPE(*CHAR) LEN(4) /* fname length */
DCL VAR(&FSNAME) TYPE(*CHAR) LEN(2000) /* Qualified file name */

/* DATA QUEUE DECLARES */
DCL VAR(&DQQ) TYPE(*CHAR) LEN(10) /* Data queue name */
DCL VAR(&DQLIB) TYPE(*CHAR) LEN(10) /* Data queue library name */
DCL VAR(&DQLEN) TYPE(*DEC) LEN(5 0) /* Total request length */
DCL VAR(&DQROP) TYPE(*CHAR) LEN(2) /* Relational operator */
DCL VAR(&DQKLEN) TYPE(*DEC) LEN(5 0) /* Key length */
DCL VAR(&DQKEY) TYPE(*CHAR) LEN(256) /* Key value */

/* REMOTE SQL DECLARES */
DCL VAR(&RSOBJ) TYPE(*CHAR) LEN(10) /* Object name */
DCL VAR(&RSLIB) TYPE(*CHAR) LEN(10) /* Object library name */
DCL VAR(&RSCMT) TYPE(*CHAR) LEN(1) /* Commitment control level */
DCL VAR(&RSMODE) TYPE(*CHAR) LEN(1) /* Block/Update mode indicator*/
DCL VAR(&RSCID) TYPE(*CHAR) LEN(1) /* Cursor ID */
DCL VAR(&RSSTN) TYPE(*CHAR) LEN(18) /* Statement name */
DCL VAR(&RSRSU) TYPE(*CHAR) LEN(4) /* Reserved */
DCL VAR(&RSREQ) TYPE(*CHAR) LEN(1925) /* SQL statement */

/* NETWORK PRINT SERVER DECLARES */
DCL VAR(&NPFMT) TYPE(*CHAR) LEN(8) /* Format name */
DCL VAR(&NPFID) TYPE(*CHAR) LEN(4) /* Function identifier */
/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT SPLF0100 */
DCL VAR(&NPJOBN) TYPE(*CHAR) LEN(10) /* Job name */
DCL VAR(&NPUSRN) TYPE(*CHAR) LEN(10) /* User name */
DCL VAR(&NPJOB#) TYPE(*CHAR) LEN(6) /* Job number */

```

```

DCL VAR(&NPFILE) TYPE(*CHAR) LEN(10)/* File name */
DCL VAR(&NPFIL#) TYPE(*CHAR) LEN(4) /* File number */
DCL VAR(&NPLEN) TYPE(*CHAR) LEN(4) /* Data Length */
DCL VAR(&NPDATA) TYPE(*CHAR) LEN(2000) /* Data */

DCL VAR(&DBNUM) TYPE(*CHAR) LEN(4) /* Number of libraries */
DCL VAR(&DBLIB2) TYPE(*CHAR) LEN(10) /* Library name */

/* DATA QUEUE SERVER DECLARES */
DCL VAR(&DQFMT) TYPE(*CHAR) LEN(8) /* Format name */
DCL VAR(&DQFID) TYPE(*CHAR) LEN(4) /* Function IDENTIFIER */
DCL VAR(&DQOOBJ) TYPE(*CHAR) LEN(10) /* Object name */
DCL VAR(&DQOLIB) TYPE(*CHAR) LEN(10) /* Library name */
DCL VAR(&DQOROP) TYPE(*CHAR) LEN(2) /* Relational operator */
DCL VAR(&DQOLEN) TYPE(*CHAR) LEN(4) /* Key length */
DCL VAR(&DQOKEY) TYPE(*CHAR) LEN(256) /* Key */

/* CENTRAL SERVER DECLARES */
DCL VAR(&CSFMT) TYPE(*CHAR) LEN(8) /* Format name */
DCL VAR(&CSFID) TYPE(*CHAR) LEN(4) /* Function identifier */
/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZSCL0100 */
DCL VAR(&CSCNAM) TYPE(*CHAR) LEN(255) /* Unique client name */
DCL VAR(&CSLUSR) TYPE(*CHAR) LEN(8) /* License users handle */
DCL VAR(&CSPID) TYPE(*CHAR) LEN(7) /* Product identification */
DCL VAR(&CSFID) TYPE(*CHAR) LEN(4) /* Feature identification */
DCL VAR(&CSRID) TYPE(*CHAR) LEN(6) /* Release identification */
DCL VAR(&CSTYPE) TYPE(*CHAR) LEN(2) /* Type of information req */
/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZSCS0100 */
DCL VAR(&CSCNAM) TYPE(*CHAR) LEN(255) /* Unique client name */
DCL VAR(&CSCMTY) TYPE(*CHAR) LEN(255) /* Community name */
DCL VAR(&CSNODE) TYPE(*CHAR) LEN(1) /* Node type */
DCL VAR(&CSNNAM) TYPE(*CHAR) LEN(255) /* Node name */
/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZSCN0100 */
DCL VAR(&CSFROM) TYPE(*CHAR) LEN(4) /* From CCSID */
DCL VAR(&CSTO) TYPE(*CHAR) LEN(4) /* To CCSID */
DCL VAR(&CSCTYP) TYPE(*CHAR) LEN(2) /* Type of conversion */
/* DATABASE SERVER DECLARES */
DCL VAR(&DBFMT) TYPE(*CHAR) LEN(8) /* Format name */
DCL VAR(&DBFID) TYPE(*CHAR) LEN(4) /* Function identifier

/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAD0100 */
DCL VAR(&DBFILE) TYPE(*CHAR) LEN(128) /* File name */
DCL VAR(&DBLIB) TYPE(*CHAR) LEN(10) /* Library name */
DCL VAR(&DBMBR) TYPE(*CHAR) LEN(10) /* Member name */
DCL VAR(&DBAUT) TYPE(*CHAR) LEN(10) /* Authority to file */
DCL VAR(&DBBFIL) TYPE(*CHAR) LEN(128) /* Based on file name */
DCL VAR(&DBBLIB) TYPE(*CHAR) LEN(10) /* Based on library name */
DCL VAR(&DBOFIL) TYPE(*CHAR) LEN(10) /* Override file name */
DCL VAR(&DBOLIB) TYPE(*CHAR) LEN(10) /* Override library name */
DCL VAR(&DBOMBR) TYPE(*CHAR) LEN(10) /* Override member name */

/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAD0200 */
DCL VAR(&DBNUM) TYPE(*CHAR) LEN(4) /* Number of libraries */
DCL VAR(&DBLIB2) TYPE(*CHAR) LEN(10) /* Library name */

/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAQ0100 */
DCL VAR(&DBSTMT) TYPE(*CHAR) LEN(18) /* Statement name */
DCL VAR(&DBCRRS) TYPE(*CHAR) LEN(18) /* Cursor name */
DCL VAR(&DBOPT) TYPE(*CHAR) LEN(2) /* Prepare option */
DCL VAR(&DBATTR) TYPE(*CHAR) LEN(2) /* Open attributes */
DCL VAR(&DBPKG) TYPE(*CHAR) LEN(10) /* Package name */
DCL VAR(&DBPLIB) TYPE(*CHAR) LEN(10) /* Package library name */
DCL VAR(&DBDRDA) TYPE(*CHAR) LEN(2) /* DRDA indicator */
DCL VAR(&DBCMT) TYPE(*CHAR) LEN(1) /* Commit control level */
DCL VAR(&DBTEXT) TYPE(*CHAR) LEN(512) /* First 512 bytes of stmt */

```

```

/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAR0100 */
DCL VAR(&DBLIBR) TYPE(*CHAR) LEN(20) /* Library name */
DCL VAR(&DBRDBN) TYPE(*CHAR) LEN(36) /* Relational Database name */
DCL VAR(&DBPKGR) TYPE(*CHAR) LEN(20) /* Package name */
DCL VAR(&DBFILR) TYPE(*CHAR) LEN(256) /* File name (SQL alias) */
DCL VAR(&DBMBRR) TYPE(*CHAR) LEN(20) /* Member name */
DCL VAR(&DBFFMT) TYPE(*CHAR) LEN(20) /* Format name */

/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAR0200 */
DCL VAR(&DBPLIB) TYPE(*CHAR) LEN(10) /* Primary key table lib */
DCL VAR(&DBPTBL) TYPE(*CHAR) LEN(128) /* Primary key table */
DCL VAR(&DBFLIB) TYPE(*CHAR) LEN(10) /* Foreign key table lib */
DCL VAR(&DBFTBL) TYPE(*CHAR) LEN(128) /* Foreign key table */

/* REMOTE COMMAND SERVER DECLARES */
DCL VAR(&RCFMT) TYPE(*CHAR) LEN(8) /* Format name */
DCL VAR(&RCFID) TYPE(*CHAR) LEN(4) /* Function identifier */
DCL VAR(&RCPGM) TYPE(*CHAR) LEN(10) /* Program name */
DCL VAR(&RCLIB) TYPE(*CHAR) LEN(10) /* Program library name */
DCL VAR(&RCNUM) TYPE(*CHAR) LEN(4) /* Number of parms or cmdlen */
DCL VAR(&RCDATA) TYPE(*CHAR) LEN(6000) /* Command string nor parms */

/* SIGNON SERVER DECLARES */

DCL VAR(&SOFMT) TYPE(*CHAR) LEN(8) /* Format name */
DCL VAR(&SOFID) TYPE(*CHAR) LEN(4) /* Function identifier */

/*****
/*
/* OTHER DECLARES
/*
/*
/*****
DCL VAR(&WRKLEN) TYPE(*CHAR) LEN(5)
DCL VAR(&DECLEN) TYPE(*DEC) LEN(8 0)
/* * * * * *
/*
/* EXTRACT THE VARIOUS PARAMETERS FROM THE STRUCTURE */
/*
/* * * * * *
/*
/* HEADER */
CHGVAR VAR(&USER) VALUE(%SST(&REQUEST 1 10))
CHGVAR VAR(&APPLIC) VALUE(%SST(&REQUEST 11 10))
CHGVAR VAR(&FUNCTN) VALUE(%SST(&REQUEST 21 10))

/* VIRTUAL PRINTER */
CHGVAR VAR(&VPOBJ) VALUE(%SST(&REQUEST 31 10))
CHGVAR VAR(&VPLIB) VALUE(%SST(&REQUEST 41 10))
CHGVAR VAR(&WRKLEN) VALUE(%SST(&REQUEST 71 5))
CHGVAR VAR(&VPLEN) VALUE(%BINARY(&WRKLEN 1 4))
CHGVAR VAR(&VPOUTQ) VALUE(%SST(&REQUEST 76 10))
CHGVAR VAR(&VPQLIB) VALUE(%SST(&REQUEST 86 10))

/* TRANSFER FUNCTION */
CHGVAR VAR(&TFOBJ) VALUE(%SST(&REQUEST 31 10))
CHGVAR VAR(&TFLIB) VALUE(%SST(&REQUEST 41 10))
CHGVAR VAR(&TFMBR) VALUE(%SST(&REQUEST 51 10))
CHGVAR VAR(&TFFMT) VALUE(%SST(&REQUEST 61 10))
CHGVAR VAR(&WRKLEN) VALUE(%SST(&REQUEST 71 5))
CHGVAR VAR(&TFLEN) VALUE(%BINARY(&WRKLEN 1 4))
CHGVAR VAR(&TFREQ) VALUE(%SST(&REQUEST 76 1925))

/* FILE SERVER */
CHGVAR VAR(&FSFID) VALUE(%SST(&REQUEST 21 4))
CHGVAR VAR(&FSFMT) VALUE(%SST(&REQUEST 25 8))
CHGVAR VAR(&FSREAD) VALUE(%SST(&REQUEST 33 1))

```

```

CHGVAR VAR(&FSWRITE) VALUE(%SST(&REQUEST 34 1))
CHGVAR VAR(&FSRDWRT) VALUE(%SST(&REQUEST 35 1))
CHGVAR VAR(&FSDLT) VALUE(%SST(&REQUEST 36 1))
CHGVAR VAR(&FSLEN) VALUE(%SST(&REQUEST 37 4))
CHGVAR VAR(&DECLEN) VALUE(%BINARY(&FSLEN 1 4))
CHGVAR VAR(&FSNAME) VALUE(%SST(&REQUEST 41 &DECLEN))

/* DATA QUEUES */
CHGVAR VAR(&DQQ) VALUE(%SST(&REQUEST 31 10))
CHGVAR VAR(&DQLIB) VALUE(%SST(&REQUEST 41 10))
CHGVAR VAR(&WRKLEN) VALUE(%SST(&REQUEST 71 5))
CHGVAR VAR(&DQLEN) VALUE(%BINARY(&WRKLEN 1 4))
CHGVAR VAR(&DQROP) VALUE(%SST(&REQUEST 76 2))
CHGVAR VAR(&WRKLEN) VALUE(%SST(&REQUEST 78 5))
CHGVAR VAR(&DQKLEN) VALUE(&WRKLEN)
CHGVAR VAR(&DQKEY) VALUE(%SST(&REQUEST 83 &DQKLEN))

/* REMOTE SQL */
CHGVAR VAR(&RSOBJ) VALUE(%SST(&REQUEST 31 10))
CHGVAR VAR(&RSLIB) VALUE(%SST(&REQUEST 41 10))
CHGVAR VAR(&RSCMT) VALUE(%SST(&REQUEST 51 1))
CHGVAR VAR(&RSMODE) VALUE(%SST(&REQUEST 52 1))
CHGVAR VAR(&RSCID) VALUE(%SST(&REQUEST 53 1))
CHGVAR VAR(&RSSSTN) VALUE(%SST(&REQUEST 54 18))
CHGVAR VAR(&RSRSU) VALUE(%SST(&REQUEST 72 4))
CHGVAR VAR(&RSREQ) VALUE(%SST(&REQUEST 76 1925))

/* NETWORK PRINT SERVER */
CHGVAR VAR(&NPFMT) VALUE(%SST(&REQUEST 21 8))
CHGVAR VAR(&NPFID) VALUE(%SST(&REQUEST 29 4))

/* IF FORMAT IS SPLF0100 */
IF COND(&NPFMT *EQ 'SPLF0100') THEN(DO)
CHGVAR VAR(&NPJOB) VALUE(%SST(&REQUEST 33 10))
CHGVAR VAR(&NPUSR) VALUE(%SST(&REQUEST 43 10))
CHGVAR VAR(&NPJOB#) VALUE(%SST(&REQUEST 53 6))
CHGVAR VAR(&NPFILE) VALUE(%SST(&REQUEST 59 10))
CHGVAR VAR(&NPFIL#) VALUE(%SST(&REQUEST 69 4))
CHGVAR VAR(&NPLEN) VALUE(%SST(&REQUEST 73 4))
CHGVAR VAR(&DECLEN) VALUE(%BINARY(&NPLEN 1 4))
CHGVAR VAR(&NPDATA) VALUE(%SST(&REQUEST 77 &DECLEN))
ENDDO

/* DATA QULUE SERVER */
CHGVAR VAR(&DQFMT) VALUE(%SST(&REQUEST 21 8))
CHGVAR VAR(&DQFID) VALUE(%SST(&REQUEST 29 4))
CHGVAR VAR(&DQOOBJ) VALUE(%SST(&REQUEST 33 10))
CHGVAR VAR(&DQOLIB) VALUE(%SST(&REQUEST 43 10))
CHGVAR VAR(&DQOROP) VALUE(%SST(&REQUEST 53 2))
CHGVAR VAR(&DQOLEN) VALUE(%SST(&REQUEST 55 4))
CHGVAR VAR(&DQOKEY) VALUE(%SST(&REQUEST 59 256))

/* CENTRAL SERVER */
CHGVAR VAR(&CSFMT) VALUE(%SST(&REQUEST 21 8))
CHGVAR VAR(&CSFID) VALUE(%SST(&REQUEST 29 4))

/* IF FORMAT IS ZSCL0100 */
IF COND(&CSFMT *EQ 'ZSCL0100') THEN(DO)
CHGVAR VAR(&CSCNAM) VALUE(%SST(&REQUEST 33 255))
CHGVAR VAR(&CSLUSR) VALUE(%SST(&REQUEST 288 8))
CHGVAR VAR(&CSPID) VALUE(%SST(&REQUEST 296 7))
CHGVAR VAR(&CSFID) VALUE(%SST(&REQUEST 303 4))
CHGVAR VAR(&CSRID) VALUE(%SST(&REQUEST 307 6))
CHGVAR VAR(&CSTYPE) VALUE(%SST(&REQUEST 313 2))
ENDDO

```



```

/* IF FORMAT IS ZSCS0100 */
IF COND(&CSFMT *EQ 'ZSCS0100') THEN(DO)
  CHGVAR VAR(&CSCNAM) VALUE(%SST(&REQUEST 33 255))
  CHGVAR VAR(&CSCMTY) VALUE(%SST(&REQUEST 288 255))
  CHGVAR VAR(&CSNODE) VALUE(%SST(&REQUEST 543 1))
  CHGVAR VAR(&CSNNAM) VALUE(%SST(&REQUEST 544 255))
ENDDO

/* IF FORMAT IS ZSCN0100 */
IF COND(&CSFMT *EQ 'ZSCN0100') THEN(DO)
  CHGVAR VAR(&CSFROM) VALUE(%SST(&REQUEST 33 4))
  CHGVAR VAR(&CSTO) VALUE(%SST(&REQUEST 37 4))
  CHGVAR VAR(&CSCTYP) VALUE(%SST(&REQUEST 41 2))
ENDDO

/* DATABASE SERVER */
  CHGVAR VAR(&DBFMT) VALUE(%SST(&REQUEST 21 8))
  CHGVAR VAR(&DBFID) VALUE(%SST(&REQUEST 29 4))
/* IF FORMAT IS ZDAD0100 */
IF COND(&CSFMT *EQ 'ZDAD0100') THEN(DO)
  CHGVAR VAR(&DBFILE) VALUE(%SST(&REQUEST 33 128))
  CHGVAR VAR(&DBLIB) VALUE(%SST(&REQUEST 161 10))
  CHGVAR VAR(&DBMBR) VALUE(%SST(&REQUEST 171 10))
  CHGVAR VAR(&DBAUT) VALUE(%SST(&REQUEST 181 10))
  CHGVAR VAR(&DBBFIL) VALUE(%SST(&REQUEST 191 128))
  CHGVAR VAR(&DBBLIB) VALUE(%SST(&REQUEST 319 10))
  CHGVAR VAR(&DBOFIL) VALUE(%SST(&REQUEST 329 10))
  CHGVAR VAR(&DBOLIB) VALUE(%SST(&REQUEST 339 10))
  CHGVAR VAR(&DBOMBR) VALUE(%SST(&REQUEST 349 10))
ENDDO

/* IF FORMAT IS ZDAD0200 */
IF COND(&CSFMT *EQ 'ZDAD0200') THEN(DO)
  CHGVAR VAR(&DBNUM) VALUE(%SST(&REQUEST 33 4))
  CHGVAR VAR(&DBLIB2) VALUE(%SST(&REQUEST 37 10))
ENDDO

/* IF FORMAT IS ZDAQ0100 */
IF COND(&CSFMT *EQ 'ZDAQ0100') THEN DO
  CHGVAR VAR(&DBSTMT) VALUE(%SST(&REQUEST 33 18))
  CHGVAR VAR(&DBCRRS) VALUE(%SST(&REQUEST 51 18))
  CHGVAR VAR(&DBSOPT) VALUE(%SST(&REQUEST 69 2))
  CHGVAR VAR(&DBATTR) VALUE(%SST(&REQUEST 71 2))
  CHGVAR VAR(&DBPKG) VALUE(%SST(&REQUEST 73 10))
  CHGVAR VAR(&DBPLIB) VALUE(%SST(&REQUEST 83 10))
  CHGVAR VAR(&DBDRDA) VALUE(%SST(&REQUEST 93 2))
  CHGVAR VAR(&DBCMT) VALUE(%SST(&REQUEST 95 1))
  CHGVAR VAR(&DBTEXT) VALUE(%SST(&REQUEST 96 512))
ENDDO

/* IF FORMAT IS ZDAR0100 */
IF COND(&CSFMT *EQ 'ZDAR0100') THEN DO
  CHGVAR VAR(&DBLIBR) VALUE(%SST(&REQUEST 33 20))
  CHGVAR VAR(&DBRDBN) VALUE(%SST(&REQUEST 53 36))
  CHGVAR VAR(&DBPKGR) VALUE(%SST(&REQUEST 69 2))
  CHGVAR VAR(&DBATTR) VALUE(%SST(&REQUEST 89 20))
  CHGVAR VAR(&DBFULR) VALUE(%SST(&REQUEST 109 256))
  CHGVAR VAR(&DBMBRR) VALUE(%SST(&REQUEST 365 20))
  CHGVAR VAR(&DBFFMT) VALUE(%SST(&REQUEST 385 20))
ENDDO

/* THE FOLLOWING PARAMETERS ADDITIONAL FOR FORMAT ZDAR0200 */
/* IF FORMAT IS ZDAR0200 */
IF COND(&CSFMT *EQ 'ZDAR0200') THEN DO
  CHGVAR VAR(&DBPLIB) VALUE(%SST(&REQUEST 33 10))
  CHGVAR VAR(&DBPTBL) VALUE(%SST(&REQUEST 43 128))

```

```

CHGVAR VAR(&DBFLIB) VALUE(%SST(&REQUEST 171 10))
CHGVAR VAR(&DBFTBL) VALUE(%SST(&REQUEST 181 128))
ENDDO

```

```

/* REMOTE COMMAND SERVER */
CHGVAR VAR(&RCFMT) VALUE(%SST(&REQUEST 21 8))
CHGVAR VAR(&RCFID) VALUE(%SST(&REQUEST 29 4))
CHGVAR VAR(&RCPGM) VALUE(%SST(&REQUEST 33 10))
CHGVAR VAR(&RCLIB) VALUE(%SST(&REQUEST 43 10))
CHGVAR VAR(&RCNUM) VALUE(%SST(&REQUEST 33 10))
CHGVAR VAR(&RCDATA) VALUE(%SST(&REQUEST 57 6000))

```

```

/* SIGNON SERVER DECLARES */
CHGVAR VAR(&SOFNT) VALUE(%SST(&REQUEST 21 8))
CHGVAR VAR(&SOFID) VALUE(%SST(&REQUEST 29 4))

```

```

/*****/
/* */
/* BEGIN MAIN PROGRAM */
/* */

```

```

CHGVAR VAR(&STATUS) VALUE('1') /* INITIALIZE RETURN +
VALUE TO ACCEPT THE REQUEST */

```

```

/* ADD LOGIC COMMON TO ALL SERVERS */

```

```

/* PROCESS BASED ON SERVER ID */
IF COND(&APPLIC *EQ '*VPRT') THEN(GOTO CMDLBL(VPRT)) /* IF VIRTUAL PRINTER */
IF COND(&APPLIC *EQ '*TFRFCL') THEN(GOTO CMDLBL(TFR)) /* IF TRANSFER FUNCTIO*/
IF COND(&APPLIC *EQ '*FILESRV') THEN(GOTO CMDLBL(FLR)) /* IF FILE SERVERS */
IF COND(&APPLIC *EQ '*MSGFCL') THEN(GOTO CMDLBL(MSG)) /* IF MESSAGING FUNCT */
IF COND(&APPLIC *EQ '*DQSRV') THEN(GOTO CMDLBL(DATAQ)) /* IF DATA QUEUES */
IF COND(&APPLIC *EQ '*RQSRV') THEN(GOTO CMDLBL(RSQL)) /* IF REMOTE SQL */
IF COND(&APPLIC *EQ '*SQL') THEN(GOTO CMDLBL(SQLINIT)) /* IF SQL */
IF COND(&APPLIC *EQ '*NDB') THEN(GOTO CMDLBL(NDB)) /* IF NATIVE DATABASE */
IF COND(&APPLIC *EQ '*SQLSRV') THEN(GOTO CMDLBL(SQLSRV)) /* IF SQL */
IF COND(&APPLIC *EQ '*RTVOBJINF') THEN(GOTO CMDLBL(RTVOBJ)) /* IF RETRIEVE OB*/
IF COND(&APPLIC *EQ '*DATAQSRV') THEN(GOTO CMDLBL(ODATAQ)) /* IF D*/
IF COND(&APPLIC *EQ '*QNPSERV') THEN(GOTO CMDLBL(NETPRT)) /* IF NETWORK PRI*/
IF COND(&APPLIC *EQ '*CNTRLSRV') THEN(GOTO CMDLBL(CENTRAL)) /* IF CENTRAL SER*/
IF COND(&APPLIC *EQ '*RMTSRV') THEN(GOTO CMDLBL(RMTCMD)) /* IF RMTCMD/DPC */
IF COND(&APPLIC *EQ '*SIGNON') THEN(GOTO CMDLBL(SIGNON)) /* IF SIGNON */

```

```

GOTO EXIT

```

```

/* * * * * * */
/* SUBROUTINES */
/* */
/* * * * * * */

```

```

/* VIRTUAL PRINTER */
VPRT:

```

```

/* SPECIFIC LOGIC GOES HERE */

```

```

GOTO EXIT

```

```

/* TRANSFER FUNCTION */
TFR:

```

```

/* SPECIFIC LOGIC GOES HERE */

```

```

GOTO EXIT

```

```

/* FILE SERVERS */
FLR:

```

```

/* SPECIFIC LOGIC GOES HERE */

```

```

    GOTO EXIT
/* MESSAGING FUNCTION */
MSG:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* DATA QUEUES */
DATAQ:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* REMOTE SQL */
RSQL:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* DATABASE INIT */
SQLINIT:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* NATIVE DATABASE */
NDB:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* DATABASE SQL */
SQLSRV:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* RETRIEVE OBJECT INFORMATION */
RTVOBJ:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* DATA QUEUE SERVER */
ODATAQ:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* NETWORK PRINT SERVER */
NETPRT:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* CENTRAL SERVER */
CENTRAL:

    /* SPECIFIC LOGIC GOES HERE */

    GOTO EXIT
/* REMOTE COMMAND/DISTRIBUTED PROGRAM CALL */
RMTCMD:

```

```

/* IN THIS CASE IF A USER ATTEMPTS TO DO A REMOTE COMMAND/DISTRIBUTED */
/* PROGRAM CALL AND HAS A USERID OF userid THEY WILL NOT BE ALLOWED TO */
/* CONTINUE. */
IF COND(&USER *EQ 'userid') THEN(CHGVAR VAR(&STATUS) VALUE('0'))

    GOTO EXIT
/* SIGNON SERVER */
SIGNON:

/* SPECIFIC LOGIC GOES HERE */

GOTO EXIT

EXIT:
ENDPGM

```

Integrating new functions into iSeries Access for Windows and iSeries Navigator

iSeries Access for Windows allows you to integrate and distribute new or changed sections of code, customized applications or new functions into the iSeries Access for Windows client. These new functions are called plug-ins and add-ins. You can include these functions with an iSeries Access for Windows installation or migration, or you can distribute them with Selective Setup. After installation, you can maintain them using Check Service Level.

Installing, uninstalling and maintaining Plug-ins

Plug-ins allow you to integrate new functions or applications into iSeries Navigator. These new functions become a separately installable component that typically add:

- Folders and objects to the hierarchy tree
- Choices to iSeries Navigator menus
- Property pages to the property sheet for a folder or object

For more information about plug-ins, and how you can use them, refer to Developing iSeries Navigator plug-ins.

Installing, uninstalling and maintaining Add-ins

Add-ins provide a convenient way for you to distribute sections of code over your network with iSeries Access for Windows. Add-ins may consists of any combination of:

- User-written programs
- Uncompressed files
- Product setup programs or installation images

All add-ins require the file ADDIN.INI to describe the add-in to the iSeries Access for Windows installation, Selective Setup and Check Service Level functions.

Note: Add-ins can provide a convenient and simple method of distributing files across your network. However, if you incorporate programs or setup programs into an add-in, consider the following add-in requirements and considerations.

Integrating Plug-ins

Plug-ins should reside on a source directory on the host. Then you can distribute the plug-in to your users as part of the installation process, or with Selective Setup. After the installation, use Check Service Level to handle upgrades.

See Distribute plug-ins for more detailed information.

Installing and uninstalling plug-ins

If the plug-in resides on the installation source, it appears as a sub-component of iSeries Navigator. If the plug-in does not exist on your installation source, use Selective Setup to install the plug-in after completing the installation. When you start Selective Setup, supply the location of the plug-in that you want to install (refer to the following table). Selective Setup will display all Plug-ins that are available for installation at the specified location. However, some optionally installable components of iSeries Access for Windows do not appear if the client and host have different versions of OS/400.

Plug-ins for iSeries Access for Windows are located in the following directories:

| Plug-ins | Location |
|-------------|---|
| IBM | (AS/400 NetServer name)\QIBM\ProdData\OpNavPlugin |
| Third party | (AS/400 NetServer name)\QIBM\UserData\OpNavPlugin |

Plug-ins for Client Access for Windows NT/95 clients are located in the following directories:

| Plug-ins | Location |
|-------------|---|
| IBM | (AS/400 NetServer name)\QIBM\ProdData\GUIPlugin |
| Third party | (AS/400 NetServer name)\QIBM\UserData\GUIPlugin |

Note: A warning message will appear if the plug-in is not explicitly supported by iSeries Access for Windows. You will still be able to install the plug-in.

Upgrading or servicing plug-ins

To update a plug-in, simply copy the updated files into the plug-ins installation source directory on the host.

Check Service Level will maintain the version of the plug-in. Every time Check Service Level starts, it checks the plug-ins installation source directory on the host to see if the plug-in requires an upgrade. If the plug-in requires an upgrade, Check Version will begin the iSeries Access for Windows Selective Setup program in a special mode. Selective Setup then updates the plug-in.

For more information, see Check Service Level.

á

Integrating add-ins

You can install and uninstall add-ins during an installation or migration, or with Selective Setup. After installing them, you can maintain them with Check Service Level. The file ADDIN.INI describes the add-in to these functions.

Installing or Uninstalling Add-ins

To include an add-in with a iSeries Access for Windows installation, place it in the pre-defined directory on the server or other installation source. The iSeries Access for Windows installation and Selective Setup will look for add-ins in the following directory:

\QIBM\UserData\Ca400\Express\Addin\

For multiple add-ins, you can include additional subdirectories.

To install the add-in:

1. Start the iSeries Access for Windows installation, or run Selective Setup.
2. Navigate through the panels and install or remove any iSeries Access for Windows components. After you install or remove the iSeries Access for Windows components, the "Install Additional Files and Programs" dialog will appear. Any add-ins that iSeries Access for Windows finds within the pre-defined directory structure will appear with a check box next to it.
3. Click to place a check next to each add-in that you want to install on your PC.
4. Navigate through any remaining panels and iSeries Access for Windows will install any add-ins that you selected.

To uninstall an add-in, start Selective Setup. After displaying the component selection dialog, Selective Setup will display all the installed add-ins, which you can select for removal.

Note: Selective Setup may not remove all parts of the add-in if the add-in contains programs that write data to the PC, install more files or write values to the registry. In these cases, you need to add a program to the add-in that iSeries Access for Windows runs before it removes the add-in files. See ADDIN.INI for instructions on adding programs to the add-in.

Upgrading or servicing Add-ins

To update an add-in, simply copy the updated files into the add-in's installation source directory on the host: \QIBM\UserData\C4400\Express\Addin.

Check Service Level will maintain the version of the add-in. Every time Check Service Level starts, it checks the add-in's installation source directory on the host to see if the add-in requires an upgrade. If the add-in requires an upgrade, Check Version will begin the iSeries Access for Windows Selective Setup program in a special mode. Selective Setup then updates the add-in.

For more information, see Check Service Level.

Note: The add-in installation source must be present when Check Service Level runs.

iSeries NetServer administration

iSeries Access for Windows takes advantage of the IBM Operating System/400 (OS/400) function: **IBM iSeries Support for Windows Network Neighborhood (iSeries NetServer)** This function allows file serving and print serving. It is available starting with OS/400 Version 4, Release 2. Previous clients, such as Client Access for Windows 95/NT, included file and print serving within the client, though not without a price. Utilizing the capabilities of iSeries NetServer, and not including this support in the iSeries Access for Windows client, allows for several advantages:

- A smaller PC client footprint.
- Background tasks and daemons are no longer necessary

iSeries Access for Windows takes advantage of iSeries NetServer for:

- Installing iSeries Access for Windows on the PC from the iSeries server
- File serving
- Print serving

For complete documentation on setting up, administering, and using the iSeries NetServer, see iSeries NetServer. This information is also accessible through the main Information Center navigation bar. To see it, select **Networking > TCP/IP > iSeries**

Restricting users with policies and application administration

iSeries Access for Windows supports two primary methods for implementing administrative control over your network: Application Administration and policies. Application Administration bases restrictions on the iSeries user profile, and is administered through iSeries Navigator. Application Administration is available in V4R3 of OS/400; however, some functions are only supported in V4R4 or later. Policies mandate configuration settings and restrictions, and can apply to both specific PCs and individual Windows user profiles. As such, they offer greater granularity than Application Administration, but are significantly more difficult to set up and administer. In order to use policies, you must download the Microsoft system policy editor and configure your PCs and iSeries server for storage, retrieval, and application of policies you set. Generally, Application Administration is preferable if all of the functions you want to restrict are Application Administration-enabled, and if the version of OS/400 being used supports Application Administration.

For V5R2, Application Administration added support for Central Settings. The Central settings support in Application Administration provides the ability to manage most of the functions iSeries Access for Windows controls through the following policy templates:

- Runtime Restrictions (caerestr.adm)
- Mandated Connection Properties (config.adm)
- configuration Policies (caecfg.adm)

For more information about Application Administration and its support for Central Settings, see *What's new for V5R2 - Application Administration*.

For more information about Application Administration, refer to *Application Administration*.

To learn about policies, refer to the following topics:

- Overview of iSeries Access for Windows policies
- Setting up your system to use policies
- iSeries Access for Windows policy list

Overview of iSeries Access for Windows policies

You can use policies to restrict users from certain actions, and to suggest or require certain configuration features. Policies can apply to Windows user profiles, and specific PCs. However, policies do not offer control over iSeries server resources, and are not a substitute for iSeries security. For a description of what you can do with policies, refer to *Types and scopes of policies*.

Policy support in your network

Policies reside on a file server. Each time users sign-on to their Windows workstation, their workstation downloads all the policies that apply to that Windows user profile. The user's PC applies the policies to the registry before the user does anything on the workstation. Each Windows operating system comes with the code needed to download policies.

To use the full capability of policies, you need the following:

- A primary logon server
- A policy server

You can use IBM iSeries Support for Windows Network Neighborhood (AS/400 NetServer) as the policy server. Windows NT/2000 and Novell Netware can be both types of servers.

See *Setting up your system to use policies* for more information.

Policy files

Policy definitions are contained in policy templates, which organize the policies into categories. iSeries Access for Windows provides five policy templates, one for each of the following functions:

- Restricting iSeries Access for Windows functions for a given system (sysname.adm)
- Restricting specific iSeries Access for Windows function at runtime (caerestr.adm)
- Restrict which components users may install or uninstall (caeinst.adm)
- Mandate or suggest configuration settings for specific environments, the systems within those environments, and some configurable values for those systems (config.adm)
- Suggest or mandate global configurable values (caecfg.adm)

You must generate the policy templates with the CWBADGEN utility before creating or modifying specific policies. Then use the Microsoft System Policy Editor to activate the templates and set their constituent policies. After setting the policies, save the changes to a policy file, for example (nt)config.pol.

Note: You must create and maintain policies for the Windows 95/98/Me and Windows NT/2000 separately. (Policies created for Windows 95 will not work on an NT system.)

See creating policies for more information.

Types and scopes of policies

Each policy iSeries Access for Windows provides is either a restriction or a configuration policy, and can address one or more scopes.

Restriction Policies

Restriction policies can usually be set to any scope and may have the following uses:

- Restrict or allow use of an iSeries Access for Windows function or action.
- Include restrictions for installing or uninstalling components, service packs, upgrades, or the entire product.
- Include several other restrictions. For example, you may restrict a certain type of data transfer upload, or you may restrict all types of data transfer uploads at once using the Prevent All Data Transfer to iSeries servers policy.
- Cause controls or options normally selectable to be hidden or "greyed-out".
- Notify the user when a restriction policy prevents a function they attempt from completing, usually by a message displayed in a console or a window.

Configuration Policies

Configuration policies can only be set to a user scope, and may have the following uses:

- Pre-configure settings that the end user could normally configure themselves.
- Configure values, features that the user may normally enable or disable, lists of environments and connections.
- "Grey-out" a mandated value. When a configuration policy mandates a value, the input field for that value will not accept changes.

Configuration policies may be either suggested or mandated.

- Suggested: The value provided will be used unless explicitly configured by the user or set by an application program. This effectively overrides the normal default value iSeries Access for Windows would use, but does not force use of the value — a new value may be specified, overriding the suggested value.
- Mandated: The value provided will be used — neither the user nor application programs may change it.

Policy Scopes

There are three scopes at which each policy may be set: machine scope, user scope and iSeries connection scope. Some policies may be set at more than one scope, while others may not.

| Scope | Description |
|--|--|
| Machine scope | A policy set at this scope applies to all users of the PC. The only exception is when the same policy is set for a specific user to override the machine scope setting. |
| User Scope | A policy set at this scope can be applied on a per-user basis. It may be set for some users, but not others. It may be set for the "Default User" (any user without an individual policy configuration) as well. Some user scope policies provide a setting that allows a function regardless of the machine scope setting. When this setting is used, the machine scope setting is ignored. |
| iSeries Connection (or "Per-System") Scope | Some policies that can be set at user or machine scope may be more narrowly set at iSeries connection scope within the user or machine scope. When set at iSeries connection scope, the policy setting is applied only when working with the named iSeries system. For example, if a restriction policy is set at iSeries connection scope inside of user scope, where the iSeries system is named SYS1 and the user is USER1, the function is restricted only when USER1 works with SYS1. <p>Note: If a policy is set at iSeries connection scope, this setting takes precedence over the user or machine scope setting. For example, if default user mode is mandated for user USER1 to be "Use default user id", but set for system SYS1 to be "Use Windows user id and password", when USER1 connects to SYS1, his Windows user id and password are used. When USER1 connects to any other system, the specified default user id is used</p> <p>Note: To enable setting policies at this scope, you must generate and use one or both of the following policy templates:</p> <ul style="list-style-type: none"> • config.adm — Configured environments and connections template • sysname.adm — Per-system (by iSeries system name) template |

Setting up your system to use policies

In order to use iSeries Access for Windows policies, complete the following steps:

1. Configure the iSeries server
2. Configure the client PC to download policies from the iSeries server
3. Create policy files

Configuring an iSeries server for policies

Use the following steps to configure your iSeries server for serving policies. These steps assume that you have Windows PCs in your network.

- Configure your iSeries server as an iSeries NetServer, if this has not already been done.
- Create an integrated file system folder to hold your policy files.

Configuring client PCs for policies

Some configuration of the client PCs in your network is required so that they will accept policy downloads from your iSeries system.

- Windows 95/98/Me systems

- Windows NT/2000/XP systems

Alternatively, if you place the policy file on the NETLOGON share on the iSeries 400 logon server, the user's PC will download the policy file automatically when the user logs onto an iSeries domain.

Configuring Windows 95/98/Me PCs for policies: Complete these steps to download and accept policies for your Windows 95/98 PC.

1. Make the iSeries NetServer accessible via TCP/IP from your PC. If you are using Domain Name Server (DNS) ensure that the iSeries NetServer name is in the hosts table on the DNS. If you are using an LMHOSTS file, be sure that you have an entry for the iSeries NetServer. Also ensure that the entry specifies the #PRE directive, for example:
9.4.3.240 QYOURSYS#PRE
2. Verify that your PC can communicate with your iSeries server.
3. Enable user profiles on the Windows desktop so that the server can provide policies for each user.
 - a. Go to **Start** → **Settings** → **Control Panel** → **Passwords**.
 - b. Select the **User Profiles** tab.
 - c. Make sure that the **Users can customize their preferences and desktop settings** button is selected.
 - d. Click **OK** and restart your computer.

Change the registry so that each Windows 95/98/Me PC on your network can download the policy file that you create. You can download a tool to do this for you. Download cwbpoluz from:

<http://www.ibm.com/eserver/iseries/access/cadownld.htm>  .

Configuring Windows NT/2000/XP PCs for policies: Each Windows NT/2000/XP workstation in your network needs to download the policy file you just created. You can download a tool that will do this for you. Download cwbpoluz from <http://www.as400.ibm.com/clientaccess/cadownld.htm>  .

Creating policy files

In order to create or modify specific policies, you will need to download the policy editor from Microsoft, generate the policy templates, and create or modify the policy file.

1. Obtain the policy editor.
2. Generate template files for iSeries Access for Windows.
3. Create your policy files.

Note: You must create and maintain policies for the Windows 95/98/Me and Windows NT/2000/XP separately. (Policies created for Windows 95/98/Me will not work on a Windows NT/2000/XP system)

Microsoft System Policy Editor: To be able to create your own policy files, you need the policy editor that Microsoft provides. The current version of the policy editor ships with Windows NT Server, the Windows NT Workstation Resource Kit, and the Office 97 Resource Kit. It is also available on the Microsoft web site. Windows 2000 requires its own version of the policy editor, which ships with Windows 2000 Server versions.

<http://www.microsoft.com> 

Search for **policy editor**. An older version of the policy editor ships on the installation CD for Windows 95. Do not use this version. It allows you to load only one policy template at a time.

Follow the directions that come with the editor to extract the file and install the policy editor and templates.

Creating iSeries Access for Windows policy templates: iSeries Access for Windows contains a program that creates the policy templates you need to control policies.

1. Open an MS-DOS window.
2. Go to the iSeries Access for Windows directory, normally located at:
[C:]\Program Files\IBM\Client Access\
3. Type the command and parameter to give you the templates for the policies that you want to set.

Policy template commands

Command `cwbadgen` with parameters

`cwbadgen /ps S1034345` (Where `s1034345` is the system name.)

`cwbadgen /std`

`cwbadgen /cfg config.adm`

Description

Generates the template for setting system specific policies, `S1034345.adm`.

Generates `caecfg.adm` (covers global configuration), `caeinrst.adm` (covers installation restrictions), & `caerestr.adm` (covers run time restrictions).

Generates the `config.adm` (configuration policy based on system configurations that exist on the PC from which this command is run). Specify the name of the file after the `/cfg` argument. In this example the template file is `config.adm`.

For more information about the `cwbadgen` utility, refer to the

Creating and updating policy files: Create policy files to control default computer or default user actions.

1. Start the policy editor by double-clicking on **poledit.exe**.
2. Go to **Options > Policy Template > Add**.
3. Go to the location where you stored the `.adm` files that you created in creating policy templates.
4. Select the `.adm` files that you want to add and press **Add**. Keep doing this until you have added all the `.adm` files that you want to use. Then click **OK**.
5. Select **File > New Policy**.
6. Set your policies and save the policy file:

`\\QYOURSYS\POLICIES\config.pol` (for Windows 95/98)

Or:

`\\QYOURSYS\POLICIES\ntconfig.pol` (for Windows NT)

Where:

- `QYOURSYS` is the name of your AS/400 NetServer.
- `POLICIES` is the name of the shared file folder on your AS/400 NetServer.
- `(nt) config.pol` is the name of your policies file.

To update the policy file, open your policy file with the policy editor, make your changes and save the file back to the above location.

Note: You must create and maintain policies for the Windows 95/98/Me and Windows NT/2000 separately. (Policies created for Windows 95 will not work on an NT system, and vice-versa.)

iSeries Access for Windows policy list

iSeries Access for Windows supports Microsoft System Policies. Administrators can use policies to control which functions and settings are available to each user. This topic lists all the policies that iSeries Access for Windows provides, and describes the effects and scope of each.

Sets of policies are defined by template files. You can generate policy templates for iSeries Access for Windows on a PC with iSeries Access for Windows installed using the **cwbadgen** command. See creating policy templates for details.

- Policies by function
Lists policies by the function they effect.
- Policies by template
Lists the templates and their associated policies.

For a general description of policies in iSeries Access for Windows, see Policy overview.

Policies by function

The following table lists iSeries Access for Windows Policies by the function they effect.

| Function | Related policies |
|---|---|
| ActiveX Automation Objects | <ul style="list-style-type: none"> • Prevent data transfer upload automation object • Prevent data transfer download automation object • Prevent remote command automation object • Prevent remote program automation object • Prevent data queue automation object |
| Communications | <ul style="list-style-type: none"> • Default user mode • TCP/IP Lookup • Port lookup mode • Require secure sockets • Prevent changes to active environment • Prevent changes to environment list • Prevent connections to systems not previously defined • Prevent use of non-mandated environments • Connection timeout |
| Data Transfer: Uploads | <ul style="list-style-type: none"> • Prevent all data transfer to an iSeries server • Prevent appending or replacing host files • Prevent Data Transfer GUI uploads • Prevent usage of RFROMPCB • Prevent autostart uploads • Prevent Excel add-in uploads |
| Data Transfer: Downloads | <ul style="list-style-type: none"> • Prevent all data transfer from an iSeries server • Prevent Data Transfer GUI downloads. • Prevent usage of RTOPCB • Prevent autostart downloads • Prevent Excel add-in downloads |
| Data Transfer: iSeries server file creation | <ul style="list-style-type: none"> • Prevent host file creation • Prevent wizard iSeries server file creation • Prevent non-wizard iSeries server file creation |
| Directory update | Prevent use of directory update |

Function

Incoming remote command

Related policies

- Allow all incoming remote commands when password caching is disabled
 - Run as system
 - Command mode
 - Cache security
 - Allow generic security
 - Generic security runs command as logged on user
- Install
- Selective Setup source directory
 - Prevent setup
 - Prevent selective setup
 - Prevent uninstall
 - Prevent check service pack level
 - Prevent install service pack
 - Prevent upgrades
 - Prevent migration of pre-V4R4M0 iSeries Access for Windows settings
 - Prevent installation of individual components
 - Prevent installation of add-ins

License management

National Language Support

Time to delay before license is released

- ANSI code page
- OEM code page
- EBCDIC code page
- Bi-directional transformation of data

ODBC

- Named data sources
- Prevent program generated data sources

OLE DB

iSeries Navigator

Passwords

Prevent OLE DB provider usage

Prevent usage of iSeries Navigator

- Warn user before iSeries password expires
- Allow caching of iSeries passwords
- Prevent iSeries Access for Windows password changes

Function

PC5250 Emulation

Related policies

- Prevent configuration of display sessions
- Prevent configuration of printer sessions
- Prevent usage of PC5250 emulator
- Maximum number of PC 5250 Sessions
- Prevent changing of .WS profiles
- Prevent menu configuration
- Prevent toolbar configuration
- Prevent multi-session configuration
- Prevent keyboard configuration
- Prevent mouse configuration
- Prevent Java applet execution
- Prevent access to macros
- Prevent profile imports in Emulator Session Manager
- Prevent profile deletion in Emulator Session Manager
- Prevent directory changes in Emulator Session Manager

PC Commands

- Cwblogon
- Cwbcfg
- Cwbback
- Cwbrest
- Cwbenv
- cwbundbs
- cwbrxd
- Wrkspfl
- wrkmsg
- wrkppt
- wrkusrj

Service

- When to check
- Delay time
- Frequency
- Copy image to PC
- Run silently
- Service path
- Autostart background service job

User Interface

Prevent creation of desktop icons

Policies by template

Use these template files to control policies. See creating policy templates for more information.

Template file

caecfg.adm

Description

Policies that suggest or mandate specific configurable values. Run cwbadgen with the /std option to generate it.

caerestr.adm

Policies that restrict specific iSeries Access for Windows functions. Run cwbadgen with the /std option to generate it.

| Template file | Description |
|---------------|--|
| config.adm | Policies that mandate configuration settings for specific environments, the systems within those environments, and some configurable values for those systems. Run cwbadgen with the /cfg option to generate it. |
| caeinrst.adm | Policies that restrict what users can install or uninstall. It also restricts other functions related to installation. Run cwbadgen with the /std option to generate it. |
| SYSNAME.adm | Policies that restrict specific iSeries Access for Windows functions for a given system. Run cwbadgen with the /ps option to generate it. |

Secure Sockets Layer administration

Secure Sockets Layer (SSL) is a popular security scheme that allows the PC client to authenticate the server and encrypts all data and requests. Use it when transferring sensitive data between clients and servers. The transfer of credit card and bank statement information are examples of client/server transactions that typically take advantage of SSL. There is an increased cost in performance with SSL because of the added encryption and decryption processing.

iSeries Access for Windows includes optionally-installable support for Secure Sockets Layer (SSL) and a way to manage key databases with **IBM Key Management**. All functions of iSeries Access for Windows can communicate over SSL except Incoming Remote Command and Ultimedia. However, on a PC using an Intel 64-bit processor, such as Itanium, only 32-bit applications and connections can use SSL. iSeries Access for Windows allows SSL communications with the iSeries server at the 128-bit level of encryption.

Beginning in v5r1, client authentication is available for PC5250.



Printed in U.S.A.