IBM i OSS Docs

IBM i OSS Docs Authors

Jan 20, 2022

CONTENTS

This repo will act as the authoritative documentation location for all things IBM i Open Source Software including PASE (Yum, Node.js/Python/Ruby/PHP, ssh etc) and all native ILE languages (C,RPG,CL etc). If you see a mistake or see a way to make some of the docs better then please issue a pull request.

What if I would like to contribute (or suggest changes) to this documentation?

This documentation is open source! Please feel free to help this documentation grow in terms of volume and usability! If you would like to see changes in this documentation, please do one of the following:

1. Submit an issue with the proposed changes. Please be as detailed as possible and include as much in "publishable" form as you can.
2. If you feel comfortable doing so, send a pull request. Start by forking the project, making your changes, and following the instructions here.

CHAPTER

ONE

CONTENTS

1.1 RPM pile for IBM i releases in standard support

1.1.1 General Information

Much of the open source technology available in the 5733-OPS product is now available in RPM form! This allows for a more seamless experience for those needing to install, manage, or use open source technologies. You can use the "yum" package manager to see the entire list of what packages are available.

Notable deliverables

- Node.js
- Python 3.6
- The 'less' utility
- git
- The 'updatedb' and 'locate' utilities (in the 'findutils' package)
- GCC 6.3.0 and many development tools such as automake, autoconf, m4, libtool, etc.
- GNU versions of many common utilities such as ls, grep, sed, awk. . . ..
- GNU Nano
- many, many more things. . . ..

1.1.2 Installation

Once you have `yum` installed, you can install, remove, and upgrade rpms easily. `yum` is even able to update and install new versions of itself! But how do you install `yum` if you don't have `yum` installed? We have a Catch-22! To get around this loop, we have provided a bootstrap installer which installs and configures yum via a different mechanism which has just enough to get `yum` and installed and working. As said before, once `yum` is installed it can update itself, so this bootstrap process is only needed once!

NOTE: Don't forget to read the Usage Notes below. They are very important!

Installing with Access Client Solutions (ACS)

NOTE: This currently requires that your PC have direct HTTPS access to the public IBM file server. If for some reason, you cannot access external sites via HTTPS, refer to steps in "Offline Install Instructions (without ACS)".

You will also need SSH connectivity from your PC to the IBM i system. Be sure to have the SSH daemon running on IBM i (`STRTCPSVR *SSHD`).

This technique does not require external Internet access from your IBM i system..

- Download the latest release of Access Client Solutions
- Access the Open Source Package Management Interface through the "Tools" Menu of ACS
- For more information, see this Technote

Online Install Instructions (without ACS Open Source Management Tool)

NOTE: This requires that your IBM i have direct FTP access to the public IBM file server from your IBM i system. Many companies now block FTP access. If that is the case, refer to steps in "Offline Install Instructions (without ACS)".

- Download bootstrap.sql to your PC
- Open ACS Run SQL Scripts and connect to the IBM i you want to install to
- Open bootstrap.sql in your Run SQL Scripts window
- Execute "Run All" via Toolbar, Menu option, or Ctrl-Shift-A
- If the result is "Bootstrapping Successful" you're all good. If not, consult /tmp/bootstrap.log.

Offline Install Instructions (without ACS)

- Download bootstrap.sh and bootstrap.tar.Z to your PC
- Transfer these two files to the `/tmp` directory on your IBM i system (via FTP, mapped network drive, scp, etc). *Make sure to transfer them in binary.*
- From a 5250 terminal run the following.

```
   QSH CMD('touch -C 819 /tmp/bootstrap.log; /QOpenSys/usr/bin/ksh /tmp/bootstrap.sh␣
↪> /tmp/bootstrap.log 2>&1')
```

- If you see message QSH005: "Command ended normally with exit status 0" in the job log you're all good. If not, consult `/tmp/bootstrap.log`.

IBM i 7.1 Install Instructions (experimental; not supported!)

- Download IBM i 7.1 version of bootstrap.sh and bootstrap.tar.Z to your PC
- Transfer these two files to the `/tmp` directory on your IBM i system (via FTP, mapped network drive, scp, etc). *Make sure to transfer them in binary.*
- From a 5250 terminal run the following.

```
   QSH CMD('touch -C 819 /tmp/bootstrap.log; /QOpenSys/usr/bin/ksh /tmp/bootstrap.sh␣
↪> /tmp/bootstrap.log 2>&1')
```

- If you see message QSH005: "Command ended normally with exit status 0" in the job log you're all good. If not, consult `/tmp/bootstrap.log`.

1.1.3 Switching from FTP to HTTP(S)

The original bootstrap used FTP to connect to the public IBM file server. This server has now had HTTP and HTTPS enabled, so you can switch to using HTTP if you prefer. This is especially useful if your corporate firewall rules disallow unsecured FTP (which many do).

The easiest way to do so is to download the new repo file and replace the one at `/QOpenSys/etc/yum/repos.d/ibm.repo`. There are numerous ways to do this, but perhaps the easiest is with `curl`:

```
curl http://public.dhe.ibm.com/software/ibmi/products/pase/rpms/ibm.repo > /QOpenSys/
↪etc/yum/repos.d/ibm.repo
```

NOTE: If you download it another way, make sure to transfer it to the IBM i system in binary or ASCII modes.

If you are switching to HTTPS (recommended), first install the CA certificates package to avoid CURL SSL errors, and then change the protocol in the repo file from HTTP to HTTPS (the original file will be saved with suffix "_backup"):

```
# Install SSL CA certificates
yum install ca-certificates-mozilla
# Install sed tool from repository
yum install sed-gnu
# Switch the protocol to https
/QOpenSys/pkgs/bin/sed --in-place='_backup' 's/http:/https:/' /QOpenSys/etc/yum/repos.
↪d/ibm.repo
```

1.1.4 Using yum on an IBM i system without internet access

If your IBM i system does not have access to the internet, yum will not be able to connect to the public IBM file server to download rpms. There are a number of options to overcome this limitation:

1. Use a proxy

If you have a proxy server available to you on your local intranet, yum can be configured to use it. Under the `main` section in `/QOpenSys/etc/yum/yum.conf` you can set `proxy`, `proxy_username`, and `proxy_password` options. eg.

```
[main]
proxy=http://proxy.mycompany.example.com:1234
proxy_username=user_name
proxy_password=passw0rd
```

2. Create a local repository mirror

If you want to keep a local cache of the repository, you can use the `reposync` and `createrepo` commands to create a complete copy of the remote repo. You'll need to do this from a system which does have external (outbound) network access. It can be on any OS which has the above tools available (on IBM i `yum install yum-utils createrepo` to install them).

```
reposync -p /path/to/repo/root -r ibm
createrepo /path/to/repo/root/ibm
```

If you are running this from a non-IBM i system, you will need to run the `reposync` command twice, specifying different `-a` parameters to download all the different architecture packages, eg.

```
reposync -a ppc64 -p /path/to/repo/root -r ibm
reposync -a fat   -p /path/to/repo/root -r ibm
createrepo /path/to/repo/root/ibm
```

Once this is done, you can share `/path/to/repo/root/ibm` on your local network using an HTTP server. This will be your new IBM i repository URL, so change the URL in the `ibm.repo` file to match.

Now that you are downloading the files through a mirror, you will need to keep the mirror in sync. Setting up a periodic task (via scheduled job or `cron`) to run the above commands will make this easy, but you can also do it manually as well.

1.1.5 Must-know Usage Notes!!! (READ THIS AFTER YOU INSTALL)

All software provided by the RPMs will install in to the `/QOpenSys/pkgs` prefix. You can fully qualify the path to the program or you can add `/QOpenSys/pkgs/bin` to your `PATH` to use the software. There are currently no plans to add symlinks in to `/QOpenSys/usr/bin` or `/QOpenSys/usr/lib`, though you can certainly do so if you like.

Fully Qualifying

```
/QOpenSys/pkgs/bin/bash --version
GNU bash, version 4.4.12(1)-release (powerpc-ibm-os400)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Adjusting your PATH

```
PATH=/QOpenSys/pkgs/bin:$PATH
export PATH

bash --version
GNU bash, version 4.4.12(1)-release (powerpc-ibm-os400)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

If you want to make your `PATH` setting permanent, add the above line to your `$HOME/.profile`. You can do this easily (from a shell) like so.

```
echo 'PATH=/QOpenSys/pkgs/bin:$PATH' >> $HOME/.profile
echo 'export PATH' >> $HOME/.profile
```

1.1.6 Installing additional software

Using Access Client Solutions (ACS)

This Technote demonstrates how to use ACS to perform simple package management tasks such as adding, removing, or upgrading software.

Yum cheat sheet

If you don't know how to use yum, Red Hat has a handy "cheat sheet" available here.

Common commands

- Install a package: `yum install <package>`
- Remove a package: `yum remove <package>`
- Search for a package: `yum search <package>`
- List installed packages: `yum list installed`
- List available packages: `yum list available`
- List all packages: `yum list all`

Installing Python 3 and some useful Python packages

```
yum install python3-pip python3-ibm_db python3-itoolkit
```

Installing Python 3 Machine Learning packages

```
yum install python3-numpy python3-pandas python3-scikit-learn python3-scipy
```

Installing Node.js

```
yum install nodejs10
```

Installing GCC and development tools

```
yum group install "Development tools"
```

Using a chroot

If you'd like to install in to a chroot, you can use the scripts from ibmichroot to set up a chroot using the `chroot_minimal.lst` and extract the bootstrap to there.

If you install to the root of the OS, you can use rpm to help install chroots. Use the `chroot_minimal.lst` to set up the chroot and then use the `--installroot` option on rpm to install the rpm in to that chroot.

```
yum --installroot=<path too chroot> install <package list>
```

The following dummy packages exist to satisfy RPM dependencies inside the chroot.

```
pase-libs-dummy-7.1-0.ibmi7.1.fat.rpm
coreutils-pase-dummy-7.1-0.ibmi7.1.ppc.rpm
```

1.1.7 Troubleshooting

Having issues? Please check out the *troubleshooting guide*.

1.1.8 FAQ

What if I run in to issues?

See **Troubleshooting** section above.

How do I get support for open source on IBM i?

Open source support is available through community channels or an IBM premium support offering. See http://ibm.biz/ibmi-oss-support

Is 5733-OPS required in order to install the RPM-based deliverables?

No. 5733-OPS does not need to be installed.

When will tools and language runtimes be 64-bit enabled?

Most of the software available in RPM form is 64-bit, including the Python and Node.js runtimes

Will 5733-OPS be updated to ship Node.js version 8, Python 3.6, or other goodies that are currently in RPM form only?

There are currently no plans to deliver these packages in the 5733-OPS installable product. If you have a business need for such, please submit an RFE with your justification.

Is this the same thing as Perzl.org or other RPM's I have heard of (or used) in the past?

No. These RPM's are not AIX RPM's. They are IBM i RPMs shipping IBM i software. Built on IBM i, for IBM i.

What if I am on an IBM i release no longer in standard support?

IBM strives to provide community open source software packages for IBM i releases in standard support. Packages (including the initial installer) that are delivered for any IBM i release no longer in standard support may be rebuilt without notice, in an effort to leverage the latest technology for IBM i customers.

IBM repositories

Information on IBM-provided repositories can be found here.

Third-party (non-IBM) repositories

Information on third-party repositories can be found here.

1.2 Common Open Source Problems (and how to fix them)

1.2.1 Things to always check first when troubleshooting

The first step in troubleshooting is to ensure that all of the following are true:

- You are accessing the system with an SSH terminal
  - If you insist on using 5250, favor QP2TERM over QSH. If using QSH, ensure that the `QIBM_MULTI_THREADED` environment variable is set to `Y` **before** launching QSH
  - If you insist on using 5250, be aware that you are more likely to have problems, even if the above precautions are taken.
- You have your `PATH` environment variable set properly. See Setting your PATH for guidance.
- You are only editing files with only ASCII or UTF8-based editors, such as:
  - (Recommended) A cross-platform editor such as VSCode, Notepad++, jEdit, etc.
  - an SSH terminal UI-based editor, such as vim, nano, or jmacs.
- You don't have CRLF line terminators in your files. This can happen if you have edited your file from Windows and do not have your editor properly configured. If you are unsure, install the `dos2unix` package and run the resulting `dos2unix` utility on your file.

1.2.2 Shell cannot find yum command

When running the `yum` command from the command line, you encounter an error like:

- `-bash: yum: command not found`
- `yum: not found`
- `ksh: yum: not found`
- `qsh: 001-0019 Error found searching for command yum. No such path or directory.`

Solution:

Add `/QOpenSys/pkgs/bin` to the beginning of your PATH environment variable. See Setting your PATH for details

Also, please don't use QSH for open source tools. Use an SSH terminal instead.

1.2.3 yum can't connect to the repository

When running yum from QSH, any commands that connect to the repository (install upgrade, etc) fail with a message like so:

```
yum install python3
https://public.dhe.ibm.com/software/ibmi/products/pase/rpms/repo/repodata/repomd.xml:␣
↪[Errno 14] curl#6 - "getaddrinfo() thread failed to start"
Trying other mirror.
Error: Cannot retrieve repository metadata (repomd.xml) for repository: ibm. Please␣
↪verify its path and try again
```

Solution:

Run yum via SSH or the ACS Open Source Package Manager GUI. These are the ideal interfaces for working with yum and the rest of the open source ecosystem.

If you need to work from 5250, QP2TERM is preferred over QSH, but QSH *will* work as long as the `QIBM_MULTI_THREADED` environment variable is set to `Y` at the job level.

1.2.4 Up arrow doesn't recall previous commands

Using arrow keys in the shell causes "garbage" to be displayed on the screen instead of cycling through command history (eg. `^[[A^[[D^[[C^[[C^[[D^[[A`)

Solution:

The default shell used by SSH is `bsh`, which is very primitive. You will probably want to set `bash` as your default shell. See Setting bash as your shell for details.

1.2.5 User input is not working properly when running in 5250

Generally speaking, **open source programs do not work well in 5250 interfaces** such as QSH or Qp2Term. This may result in improper processing of control keys, phantom user input from previous commands, "garbage" characters printed to the screen, or a host of other issues.

Solution:

Please use an SSH terminal emulator and an SSH connection. Also, for usability, you probably want to set `bash` as your default shell. See Setting bash as your shell for details.

1.2.6 Python can't find packages installed from ACS

After installing a Python package from ACS (eg. `python3-Pillow`), it can't be found.

Solution:

- Ensure you are running the correct version of Python for the package that was installed: `python3` for packages with the `python3` prefix and `python2` for packages with the `python2` prefix.
- Also ensure that your `PATH` environment variable is set to find `/QOpenSys/pkgs/bin` before `/QOpenSys/usr/bin` and `/usr/bin`, especially if you potentially have other Pythons installed from 5733-OPS or Perzl rpms. See Setting your PATH for details.

NOTE: neither `python` rpm installed via yum creates a `python` symlink, so you cannot just run `python`.

1.2.7 Yum or RPM fails with "Error: Error: rpmdb open failed"

When Running a `yum` or `rpm` command you encounter

```
Error: Error: rpmdb open failed
```

Solution:

The rpm database has gotten corrupted. Please report this issue here.

The common solution is to rebuild the database with

```
/QOpenSys/pkgs/bin/rpm --rebuilddb
```

1.2.8 Yum or RPM fails with "db4 error(19) from dbenv->open: The specified device does not exist."

Running yum you encounter an error like

```
error: db4 error(19) from dbenv->open: The specified device does not exist.
error: cannot open Packages index using db4 - The specified device does not exist.␣
↪(19)
error: cannot open Packages database in /QOpenSys/var/lib/rpm
CRITICAL:yum.main:
Error: rpmdb open failed
```

You probably have journalling on for an IFS directory that rpm is using. rpm uses `mmap` to open its database files, which is incompatible with journaling.

Note: When an ILE a application tries to `mmap` an IFS file which is being journaled it gets an error - `ENOTSUP` (operation not supported), however this gets mapped to PASE as `ENODEV` (no such device) which makes things confusing.

Solution:

Ensure that journaling is disabled/omitted for `/QOpenSys/var/lib/rpm` or any subdirectory. You can use option 8 from `WRKLNK` to view the journaling attributes of a given file or directory.

1.2.9 Commands are failing in QSH

Many commands will fail in QSH for many reasons! However, a common reason is related to the QSH behavior that disallows multithreaded applications by default. The resulting error message may or may not be descriptive, but here are some examples.

git:

```
error: cannot create async thread: Resource temporarily unavailable
fatal: fetch-pack: unable to fork off sideband demultiplexer
```

node/npm:

```
[335708]: ../src/node_platform.cc:61:std::unique_ptr<unsigned int> node::Work
↪erThreadsTaskRunner::DelayedTaskScheduler::Start(): Assertion `(0) == (uv_thr
↪create(t.get(), start_thread, this))' failed.
qsh: 001-0078 Process ended by signal 5.
```

curl:

```
curl: (6) getaddrinfo() thread failed to start
```

java (openjdk):

```
Error: Port Library failed to initialize: -1
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

Solution:

As mentioned earlier, the optimal solution is to connect with an SSH client. If you insist on using 5250, favor QP2TERM over QSH. If using QSH,you must ensure that the `QIBM_MULTI_THREADED` environment variable is set to `Y` **before** launching QSH.

1.2.10 I'm still having issues

If you are having an issue that's not listed above or the solution provided did not help, please open a ticket here.

1.3 Node.js usage notes

All things assume you have PATH set correctly

1.3.1 Choosing which major version of Node.js to use

There are two ways to install Node.js:

1. Through the package manager (recommended for production. Note that there are different packages for different major versions. A simple `yum upgrade` will not upgrade any existing versions to a new major version.
2. Using *nvm*

Consult the Node.js release schedule for guidance on which version of Node.js to use. Generally speaking:

- Only even-numbered major releases should be deployed to production. The odd-numbered releases are feature releases designed to allow early access to new features. Feature releases lack both stability and a long-term support (LTS) schedule
- Choose a version that is in "Active LTS" status for production applications. "Current" status is sometimes acceptable as well, as long as the release will be entering LTS soon.
- **NEVER** run an out-of-support release
- Ensure you have a plan to upgrade before the version's "end of life" date.

1.3.2 Setting the Node.js major version

- To switch the default version of Node.js for all users, use the `/QOpenSys/pkgs/bin/nodever` utility. For instance, for Node.js version 10 to be the default, run `/QOpenSys/pkgs/bin/nodever 14`
- If you need to explicitly invoke a specific major version of Node.js, the executable is found at `/QOpenSys/pkgs/lib/nodejs<version>/bin/node`, where `<version>` is the major version. For instance, to run Node.js version 10, one could run `/QOpenSys/pkgs/lib/nodejs10/bin/node`
- To switch the default version of Node.js for a specific user, place `/QOpenSys/pkgs/lib/nodejs<version>/bin` at the beginning of the user's PATH environment variable, similar to what's documented here. For instance, that user could run the following from the shell to set their default to version 16:

```
echo 'PATH=/QOpenSys/pkgs/lib/nodejs16/bin:/QOpenSys/pkgs/bin:$PATH' >> $HOME/.profile
echo 'export PATH' >> $HOME/.profile
```

(if using `bash` as the shell, the user may need to run `hash -r`)

1.3.3 Globally-installed modules

- To use `node-gyp` (or other globally-installed modules) from the command line, follow the same instructions in the previous section to add the version-specific directory to your `PATH`

1.3.4 Modules for accessing Db2, RPG, CL etc

- Be sure to use the `itoolkit` package from npm (`npm install itoolkit`) for accessing RPG, CL, etc.
- For database access, install one of the following packages from npm: `idb-connector`, `idb-pconnector`, `odbc` (see *the ODBC doc* for further guidance on ODBC)

1.3.5 Coming from 5733-OPS?

- Any globally-installed modules must be reinstalled. Note, however, that global installations of node modules are generally considered bad practice (with a few exceptions, like `node-gyp`).

1.3.6 Using Node Version Manager (nvm)

Step 1: Install developer tools

From an SSH terminal, run:

```
yum group install "Development Tools"
yum install gcc10\*
```

Step 2: Install NVM

First, install necessary prerequisites using *yum* to verify that you have curl and/or wget installed. Make sure you set your PATH to utilize the new open source technology.

Then, simply follow the installation steps on the nvm project page

Step 3: Configure your $HOME/.nvmrc file

Create a file at `$HOME/.nvmrc`, with the following contents

```
--dest-cpu=ppc64
--without-snapshot
--shared-openssl
--without-perfctr
```

Step 4: Set necessary environment variables

Set the following environment variables:

```
OBJECT_MODE=64
CC=gcc-10
CXX=g++-10
```

Preferrably, set these in your `$HOME/.profile` and/or `$HOME/.bash_profile` (depending on your shell settings), by adding the following lines:

```
OBJECT_MODE=64
export OBJECT_MODE
CC=gcc
export CC
CXX=g++
export CXX
```

Step 5: Run nvm

NVM, like many open source commands, are best run from an SSH terminal. See the nvm project page for example usages of the command. Some examples include:

- `nvm install stable` : install the latest stable version
- `nvm install --lts` : install the latest LTS release
- `nvm use --lts` : use the latest LTS release

1.4 Python usage notes

All things assume you have PATH set correctly

1.4.1 Which versions are available?

Currently, Python versions 3.6 and 3.9 are available, via the `python3` and `python39` packages, respectively.

Important Note Performing a `yum upgrade` operation will not upgrade currently-installed versions of Python to a new major version. This would break existing apps.

1.4.2 Python virtual environments (highly recommended!)

1. Installing system-wide packages

Use `yum` to install any RPM-provided packages that you may use inside your virtual environment.

2. Creating a virtual environment

If using Python 3.9

```
python3.9 -m venv --system-site-packages /path/to/venv
```

If using Python 3.6

```
python3.6 -m venv --system-site-packages /path/to/venv
```

3. Entering and using the virtual environment

To "enter" a virtual environment, run:

```
source /path/to/venv/bin/activate
```

Once completed, you should be able to run the `python` and `pip` commands. They will use the version of Python that was used to create the virtual environment.

1.4.3 Which Python command to use outside of a virtual environment?

Python interpreter

Use the versioned python command when feasible. It is always best to run the fully-versioned python command, which is either `python3.9` or `python3.6`, since the `python` or `python3` commands may render different results depending on what packages are installed or what the current environment contains.

Alternatives if `python39` is not installed Use `python3.6`. One can alternatively use the shorthand `python3`, but including the major and minor version in the python command is more explicit and is best practice. For instance, launch your python program via:

```
python3.6 myprogram.py
```

Alternatives if `python39` is installed Use `python3.9` or simply `python`, since the `python39` package creates a `python` via `update-alternatives`.

```
python3.9 myprogram.py
```

Preferred Installer for Python (pip)

Invoke the Python interpreter command (above), followed by `-m pip`. For instance:

```
python3.9 -m pip install --upgrade xlsxwriter
```

1.4.4 Installing Python packages

See Installing Python Packages

1.4.5 Modules for accessing Db2, RPG, CL etc

- Be sure to use the `itoolkit` package from PyPI (installable via pip) for accessing RPG, CL, etc.
- For database access with `odbc`, install `python3-pyodbc` or `python3.9-pyodbc, depending on which version of Python you are using (see *the ODBC doc* for further guidance on ODBC).
- For database access with `ibm_db`, install `python3-ibm_db` or `python39-ibm_db`, depending on which version of Python you are using. **Important Note** Do not install the `ibm_db` package via pip! This package will not work when running on IBM i.

1.5 ODBC

Open Database Connectivity (ODBC) is a standardized API for connecting to databases that is cross-platform and cross-DBMS (Database Management System). By installing an general ODBC driver manager and a driver for your specific DBMS, the code that interacts with the ODBC API can be generalized. This makes ODBC a very powerful tool for both development and for production, as the application can be deployed on any system and talk to any DBMS as long as it has the correct drivers installed.

1.5.1 Contents

Why ODBC

For the open-source software team, ODBC is the preferred method of connecting to Db2 on i. There are many resons for this, including:

- Because ODBC is a technology that is used for more than just IBM i, there are many applications and technologies that are already enabled to use ODBC. Nearly all open-source programming languages (and many non-open-source languages) have some way to connect to databases through an ODBC interface, facilitating interaction with any database that has an ODBC driver (including IBM i).
- Similarly, because ODBC connectors have already been developed for so many languages and frameworks, the IBM i Open-Source Software Team doesn't have to spend time creating specific Db2 for i connectors for every new technology we deliver on the platform. This means that we can spend more time delivering new software for you and pushing what is possible on IBM i. In the future, most of the packages we develop will require that you use ODBC connections.
- As already mentioned, ODBC is useful if you want to connect to Db2 on i from off-system. Unlike CLI-based connectors, which can only be built on IBM i, ODBC connections can be created from Windows and Linux machines as well. This means that you can develop your applications on one system and then move them to IBM i when you are ready to deploy them. It also means that you can have the same application running on multiple different platforms that can all communicate with Db2 on i in the same way.
- Finally, there are many more connection options available for ODBC than on CLI. When you create an ODBC connection through a DSN or a connection string, there are approximately 70 different connection options that can be set. This includes everything from specifying the system, your username, or your password, to defining default libraries and schemas or whether or not stored procedures can be called. A full list of options can be found on the "Connection string keywords" page of the 7.4 documentation.

IBM i Access ODBC Installation

The instructions for installing and ODBC driver and manager and the IBM i ODBC driver for Db2 on i will depend on what operating system you are running. Select your operating system below to see setup instructions for getting ODBC on your system and connected to IBM i.

- *IBM i*
- *Linux*
- *Windows*
- *macOS*

IBM i

Driver Manager for IBM i

On IBM i, we will be using unixODBC as our driver manager. Fortunately, unixODBC is automatically pulled in when you install the IBM i Access ODBC Driver for Linux, so there isn't any set up that you have to do for this stage. If you want to develop applications using ODBC packages like `odbc` for Node.js, you will have to use yum to manually install `unixODBC-devel` as well.

Driver for IBM i

To get both the unixODBC driver manager and the driver that allows ODBC to talk to Db2 for i, you will have to install the ODBC driver that allows your IBM i machine to use unixODBC to talk to Db2. To get the driver, visit the IBM i Access Client Solutions page and select **Downloads for IBM i Access Client Solutions**. After logging in and redirected to the IBM I Access Client Solutions download page, select the `Download using http` tab then scroll down and download the **ACS PASE App Pkg**. More complete instructions on how to download this driver can be found at this TechNote on the ODBC Driver for the IBM i PASE environment.

When the driver has been downloaded and unzipped and transferred to your IBM i system, you can run the rpm with yum the same way you would otherwise, but giving it the location of the file instead of the name of the package:

```
yum install <package-location>/ibm-iaccess-<version>.rpm
```

This will install the Db2 ODBC driver onto your IBM i system. It will also create a driver entry in your `odbcinst.ini` and a DSN in `odbc.ini` for your local system called `*LOCAL`. This is discussed below.

Linux

Driver Manager for Linux

On Linux, we will be using unixODBC as our driver manager. Fortunately, unixODBC is automatically pulled in when you install the IBM i Access ODBC Driver for Linux, so there isn't any set up that you have to do for this stage. If you want to develop applications using ODBC packages like pyODBC for Python or odbc for Node.js, you will have to manually use yum to install `unixODBC-devel` as well.

Driver for Linux

To get both the unixODBC driver manager and the driver that allows ODBC to talk to Db2 for i, you will have to install the IBM i Access ODBC Driver for Linux. To get the driver, visit the IBM i Access Client Solutions page and select **Downloads for IBM i Access Client Solutions**. After logging in and redirected to the IBM I Access Client Solutions Download page, select the `Download using http` tab then scroll down and download the **ACS Linux App Pkg**.

In this package, there is a README that will help explain how to install the driver with either with RPMs or DEBs, depending on your Linux distribution. Just know that when you install the driver, it should pull in all of the packages you need to create an ODBC connection to Db2 for i from your Linux system.

Windows

Driver Manager for Windows

Windows comes preinstalled with an ODBC driver manager. To access it, search for `Administrative Tools` on your system (either through the search bar, or `Control Panel > System and Security > AdministrativeTools`), and then from there select ODBC Data Sources (either 32-bit or 64-bit).

From this application, you can set up your drivers.

Driver for Windows

You will have to install the ODBC driver that allows Windows ODBC driver manager to talk to Db2 on i. To get the driver, visit the IBM i Access Client Solutions page and select **Downloads for IBM i Access Client Solutions**. After logging in and redirected to the IBM I Access Client Solutions download page, scroll down and download the **ACS Windows App Pkg English (64bit)**.

When the package has been downloaded and has been installed on your system, it should be available to see on your ODBC Data Source Administrator application.

macOS

Driver Manager for macOS

On macOS, you will need unixODBC as your ODBC driver manager. Many macOS ODBC programs use another driver manager called **iodbc**, but *the IBM i ODBC driver will not work with iodbc*. unixODBC is available on `homebrew`, and can be installed running the following command:

```
brew install unixodbc
```

Driver for macOS

You will also have to install the macOS ODBC driver that allows unixODBC to talk to Db2 on i. To get the driver, visit the IBM i Access Client Solutions page and select **Downloads for IBM i Access Client Solutions**. After logging in and redirected to the IBM I Access Client Solutions download page, scroll down and download the **ACS Mac App Pkg**. The package will include a standard macOS installer package, which can be installed by double clicking orby running the `pkgutil` command.

Using ODBC

Now that you have the IBM i Access ODBC Driver installed on your system, you are ready to connect to Db2 on i.

Connection Strings

ODBC uses a connection string with keywords to create a database connection. Keywords are case insensitive, and values passed are separated from the keyword by an equals sign ("=") and end with a semi-colon (";"). As long as you are using an ODBC database connector, you should be able to pass an identical connection string in language or technology and be confident that it will correctly connect to Db2 on i. A common connection string may look something like:

```
DRIVER=IBM i Access ODBC Driver;SYSTEM=my.ibmi.system;UID=foo;PWD=bar;
```

In the above example, we define the following connection options:

- DRIVER: The ODBC driver for Db2 for i that we are using to connect to the database (and that we installed above)
- SYSTEM: The location of your IBM i system, which can be its network name, IP address, or similar
- UID: The User ID that you want to use on the IBM i system that you are connecting to
- PWD: The password of the User ID passed above.

These are only some of the over 70 connection options you can use when connecting to Db2 on i using the IBM i Access ODBC Driver. A complete list of IBM i Access ODBC Driver connection options can be found at the IBM Knowledge Center: Connection string keywords webpage. If passing connections options through the connection string, be sure to use the keyword labeled with **Connection String**.

DSNs

As you add more and more options to your connection string, your connection string can become quite cumbersome. Luckily, ODBC offers another way of defining connection options called a DSN (datasource name). Where you define your DSN will depend on whether you are using Windows ODBC driver manager or unixODBC on Linux or IBM i.

Configuration with UnixODBC (IBM i, Linux, macOS)

IBM i, Linux distributions, and macOS use unixODBC and have nearly identical methods of setting up your drivers and your DSNs.

`odbc.ini` and `.odbc.ini`

When using unixODBC, DSNs are defined in `odbc.ini` and `.odbc.ini` (note the `.` preceding the latter). These two files have the same structure, but have one important difference:

- `odbc.ini` defines DSNs that are available to **all users on the system**. If there are DSNs that should be available to everyone, they can be defined and shared here. Likely, this file is located in the default location, which depends on whether you are on IBM i or Linux:
- IBM i: `/QOpenSys/etc/odbc.ini`
- Linux: `/etc/unixODBC/odbc.ini`

If you want to make sure, the file can be found by running:

```
odbcinst -j
```

- `.odbc.ini` is found in your home directory (`~/`) and defines DSNs that are available **only to you**. If you are going to define DSNs with your personal username and password, this is the place to do it.

In both `odbc.ini` and `.odbc.ini`, you name your DSN with `[]` brackets, then specify keywords and values below it. An example of a DSN stored in `~/.odbc.ini` used to connect to an IBM i system with private credentials might look like:

```
[MYDSN]
Description             = My IBM i System
Driver                  = IBM i Access ODBC Driver
System                  = my.ibmi.system
UserID                  = foo
Password                = bar
Naming                  = 0
DefaultLibraries        = MYLIB
TrueAutoCommit          = 1
```

(**Note:** The name of the driver specified in the `Driver` keyword must match the name of a driver defined in `odbcinst.ini`. The location of this file can also be found by running `odbcinst -j` in PASE. When you install the IBM i Access ODBC Driver on your system, it automatically creates a driver entry of `IBM i Access ODBC Driver` in `odbcinst.ini`, which you should use for all IBM i connections).

When installing the IBM i Access ODBC Driver on IBM i, the driver will automatically create a DSN called `[*LOCAL]` in your `odbc.ini`:

```
### IBM provided DSN - do not remove this line ###
[*LOCAL]
Description = Default IBM i local database
Driver      = IBM i Access ODBC Driver
System      = localhost
UserID      = *CURRENT
### Start of DSN customization
### End of DSN customization
### IBM provided DSN - do not remove this line ###
```

When using this DSN, the user credentials used will be `*CURRENT`, which is the user who is running the process that is trying to connect to the ODBC driver. Use of this `*CURRENT` behavior is dependent on some server PTFs:

- 7.2: SI68113
- 7.3: SI69058
- 7.4: (none, comes with the operating system)

Like connection string keywords, DSN keywords can be found at the IBM Knowledge Center: Connection string keywords webpage. When passing connection options through a DSN, be sure to use the keyword labeled with **ODBC.INI**.

Configuration on Windows

When you have the driver installed on your system, you can now configure your datasource names (DSNs) that allow you to wrap all of your connection settings in one place that can be used by any ODBC application.

In ODBC Data Source Administrator, you can define either User DSNs or System DSNs. A User DSN will be available only to your Windows user, while a System DSN will be available to everyone. Furthermore, System DSNs must be defined per-architecture, while User DSNs are architecture agnostic.

To create a DSN, select either User DSN or System DSN and then select `Add` on the right-hand menu. It will prompt you to select a driver, and you will select `IBM i Access ODBC Driver`. Use the GUI to add configuration options, such as your username and passwords, threading, default library, and so on.

Using Your DSN

Once you have DSNs defined with the connection options you want, you can simply pass a connection string to your ODBC connections that references the DSN:

```
DSN=MYDSN
```

This will look through your DSNs for a match, and pull in all connection options defined therein. This helps keep your connection string much more manageable, and also keeps your connections string more secure, as you don't have to explicitly pass your password in plain text.

Additional options can be added to your connection string even if you use a DSN. In this way, you can extend your DSNs with options that make sense for a given use. To add more options, simply list them as you would any normal connection string:

```
DSN=MYDSN;DBQ=MYLIB,OTHERLIB;CCSID=1208;
```

Note that connection keywords specified on the connection string will override a DSN keyword that has the same functionality (e.g. a `CMT` value on the connection string will override any `CommitMode` defined for a DSN used in that connection string).

Node.js Example

This quick example will demonstrate development on a non-IBM i machine against Db2 on i, and then how you would transfer that same code to run on IBM i when you are ready to run in production.

For this example, we will be using Node.js and the `odbc` package available on NPM. Node.js is simply used as an example technology, and this same thing could be done with PHP, Python, R, or any other package that has the ability to connect to the ODBC driver manager.

Setting Up Your Development Environment

DSNs

The following instructions assume you have set up your ODBC environment *as outlined in the sections above*. On your development machine, define a private DSN similar to the following, adding in your system and user credentials:

```
[MYDSN]
Description             = The IBM i System
Driver                  = IBM i Access ODBC Driver
System                  = PUT.YOUR.SYSTEM.HERE
UserID                  = USERNAME
Password                = PASSWORD
```

Node.js

To run through this example, you will need to have Node.js installed. You can find the downloads at the official Node.js website or through your system's package manager.

When you have Node.js installed, navigate to a new folder to contain your project and run:

```
npm init -y
```

This will create a file for you called `package.json`, which tracks software you download from npm (among other things).

Next, install the `odbc` package, which allows Node.js to talk to your driver manager.

```
npm install odbc
```

You now have everything you need to connect to Db2 for i from your development machine!

Development

Using the `odbc` package, you can use a connection string that only references the DSN you defined above. Once you have the connection created, all of your queries will be run against the IBM i system defined in the DSN.

`app.js`

```
const odbc = require('odbc');

odbc.connect('DSN=MYDSN', (error, connection) => {
  if (error) { throw error; }
  // now have an open connection to IBM i from any Linux or Windows machine
```

(continues on next page)

(continued from previous page)

```
  connection.query('SELECT * FROM QIWS.QCUSTCDT', (error, result) => {
    if (error) { throw error; }
    console.log(result);
  })
});
```

Transfer to IBM i

When you are ready to transfer your program to IBM i, you just need to make sure you have everything set up on that system.

DSNs on IBM i

Like on your development machine, you will have to install your driver manager and driver. Steps to do that can be found in *installation on IBM i* section. That section will also cover instructions for downloading the Db2 for i driver and how to configure your DSNs, though this example will use the default `*LOCAL` DSN.

Node.js on IBM i

Because we want to transfer our Node.js application to our IBM i system, we will have to have Node.js installed:

```
yum install nodejs10
```

You will then have Node.js v10 on your system. You simply need to move your code over to your IBM i system. Because we want to connect to the local database, we change our DSN to be `*LOCAL` instead of `MYDSN`:

app.js

```
const odbc = require('odbc');

odbc.connect('DSN=*LOCAL', (error, connection) => {
  if (error) { throw error; }
  // now have an open connection to IBM i from any Linux or Windows machine
  connection.query('SELECT * FROM QIWS.QCUSTCDT', (error, result) => {
    if (error) { throw error; }
    console.log(result);
  })
});
```

1.6 Java

Here you'll find information about using open source Java on IBM i.

1.6.1 Contents

Java 11 Early Access RPM

A Java RPM is now available, via the `openjdk-11-ea` packages! Note, however, this is not considered generally available (GA). It is an early access package ONLY.

This page is here to address some frequently asked questions you may have.

Is this Java production-ready?

No! These are early access RPMs, denoted with the `-ea` suffix in the package name.

Why is this only "early access"?

Currently, this Java runtime has not yet completed all testing and is therefore not certified as production ready. We have, however, done thorough functional testing of various workloads, including Apache Camel, ActiveMQ, Tomcat, maven, Jenkins, etc. We are releasing an "early access" version to solicit feedback from the IBM i community, and we welcome your feedback for any testing you do!

How do I install this?

Simply use `yum` or the Open Source Package Management tool in Access Client Solutions to install the `openjdk-11-ea` package.

Where does this install?

The Java Runtime Environment (JRE) gets installed to `/QOpenSys/pkgs/lib/jvm/openjdk-11`

How do I select this JVM to run my program?

If invoking Java directly from the command line, you will need to fully-qualify the `java` binary (`/QOpenSys/pkgs/lib/jvm/openjdk-11/bin/java`) or add the executable's directory to the beginning of your PATH, like so:

```
PATH=/QOpenSys/pkgs/lib/jvm/openjdk-11/bin:$PATH
export PATH
```

If you'd like common Java-based tools, such as activemq or maven, to use this Java, you should also set the `JAVA_HOME` environment variable:

```
JAVA_HOME=/QOpenSys/pkgs/lib/jvm/openjdk-11
export JAVA_HOME
```

Why do I get error "Directory /QOpenSys/pkgs/lib/jvm/openjdk-11 in the JAVA_HOME environment variable does not contain a Java Virtual Machine."?

The default java launchers do not currently accommodate this open source distribution. This will likely be investigated and addressed in a future release. In the interim, launch Java explicitly as documented in the previous section.

Can this integrate with RPG via the *JAVA declaration?

No. This will likely be investigated for a future release.

Can I use ILE native methods?

No, and there are no plans to support this. Instead, use the JTOpen (jt400) library's `ProgramCall` support. This also means that the `*CURRENT` special value cannot be used for authentication with the JTOpen library.

Does this show up in WRKJVMJOB?

No. This will likely be added in a future release.

Can this run inside of a chroot container?

Yes!! Unlike JV1 flavors of Java, this open source version is container-friendly.

Are there any functions known not to work?

As previously stated, not all testing has been completed, and this is not considered production-ready. That being said, the `sun.nio.ch.AixAsynchronousChannelProvider` class will malfunction and cause program crashes on IBM i 7.2. On IBM i 7.3, this function requires PTF SI71837. For IBM i 7.4, PTF SI72224 is required.

Open Source Java Application Servers

This page documents open source Java application servers. All items on this page are known to work on IBM i.

Application Server	Governed By	IBM Support Available?
Apache Tomcat	The Apache Software Foundation (ASF)	yes
Wildfly	Red Hat	yes
OpenLiberty	IBM	no
Eclipse Jetty	The Eclipse Foundation	yes

Deployment Guides

- WildFly
- TomCat

1.7 Kafka

1.7.1 Streaming data to Kafka from IBM i

There are several approaches to streaming data from Db2 transactions to Kafka, including but not limited to:

1. Db2 triggers and Apache Camel: stream events in real-time
2. Kafka Connect JDBC Source connector: Simple, polling-based technique for importing Db2 data into Kafka
3. InfoSphere Data Replication and the CDC Replication Engine for Kafka (external link): a CDC-based approach that may be a good options for current IBM CDC customers
4. Native ILE Kafka client (unsupported) (external link): call Kafka functions directly from ILE programs.

1.7.2 Deploying Kafka on IBM i

These steps walk you through installing Kafka 2.6.0 (built with Scala 2.13) and deploying on an IBM i system using OpenJDK Early Access Builds.

This assumes you are using an SSH terminal and that you have your PATH set up properly, for instance:

```
PATH=/QOpenSys/pkgs/bin:$PATH
export PATH
```

and you are hopefully, but optionally, using bash

Also, note that this deploys with the default "out of the box" settings for Zookeeper and Kafka. Please refer to the Zookeeper and Kafka documentation to learn about customizing these appropriately for a production deployment as needed.

1. Download requisite software

```
yum install wget ca-certificates-mozilla gzip tar-gnu openjdk-11 coreutils-gnu sed-
↪gnu grep-gnu
```

2. Change to your installation directory

```
cd /home/myusr/mydir
```

3. Download kafka

```
wget https://dlcdn.apache.org/kafka/3.0.0/kafka_2.13-3.0.0.tgz
```

(you may need to update the version number on this and subsequent steps based on the latest version.

4. extract Kafka

```
tar xzvf kafka_2.13-3.0.0.tgz
```

5. Set up environment to use OpenJDK

```
JAVA_HOME=/QOpenSys/pkgs/lib/jvm/openjdk-11
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
```

6. Start a Zookeeper server

```
cd kafka_2.13-3.0.0/config
../bin/zookeeper-server-start.sh zookeeper.properties
```

7. Open a new session and change to your installation directory from earlier

```
cd /home/myusr/mydir
```

8. Set up environment to use OpenJDK and start a Kafka server

```
JAVA_HOME=/QOpenSys/pkgs/lib/jvm/openjdk-11
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
cd kafka_2.13-2.6.0/config
../bin/kafka-server-start.sh server.properties
```

1.8 Starting a Kafka visualizer

The Kafka visualizer used for the interactive workshop is an open source, Java-based visualizer that is accessed through a web browser. The project page has more information. It can be installed with docker via

```
docker pull kbhargava/kafka-visuals
```

and run like this (substitute host names and port numbers as appropriate).

```
docker run -p 8080:8080 --rm kbhargava/kafka-visuals idevphp.idevcloud.com:2181␣
↪idevphp.idevcloud.com:9092 DEV
```

If you want to run this visualizer on IBM i, install via yum:

```
yum install ca-certificates-mozilla
yum install https://github.com/ThePrez/kafka-visualizer/releases/download/v1.0.0/
↪kafka-visualizer-1.0.0-0.ibmi7.3.ppc64.rpm
```

... and run using the following command and navigate to `localhost:8080` on your browser:

```
/opt/kafka-visualizer/bin/kafka-visualizer
```

You can opt to use any kafka visualizer you'd like. Kafka even comes with a single-topic visualizer that can run in your SSH terminal, for instance

```
JAVA_HOME=/QOpenSys/pkgs/lib/jvm/openjdk-11
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
cd kafka_2.13-3.0.0/config
../bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic mytopic
```

1.9 Apache Camel

Apache Camel is a powerful Java-based integration framework that can be used to integrate countless technologies, including IBM i. For a brief introduction, see this article: Integrate IBM i With Anything Using Apache Camel.

Examples Examples of using Apache Camel on IBM i can be found in the IBM i OSS examples repository, in the 'camel' directory.

1.9.1 Using the Camel JT400 Component

The JT400 Component can be used to integrate with IBM i in several ways, including:

- Message queues
- Data queues
- Program call

1.9.2 Other Components

Most other Apache Camel components are platform-agnostic and can be used from IBM i or to directly interact with IBM i. Some of the components most commonly used explicitly for IBM i integration include, but are not limited to:

- JDBC/SQL
- Paho (mqtt/IoT)
- Exec (execute a command)
- File
- FTP/sFTP
- HTTP
- SSH

1.10 NGINX

1.10.1 Summary

Here you will find general information as it relates to Nginx on IBM i. This is meant to be a supplement to the official Nginx documentation.

1.10.2 Install

```
yum install nginx
```

1.10.3 Config

The following can be placed in file `/www/nginx/nginx.conf`.

```
pid /www/nginx/nginx.pid;
events {}
http {
  server {
    listen       8001;
    server_name  localhost;
    location / {
      root   /www/nginx/html;
      index  index.html;
    }
  }
}
```

1.10.4 Starting and Stopping

To start Nginx you need use use the `/QOpenSys/pkgs/bin/nginx` binary and the `-c` option to declare where the config location. This is like submitting a job to batch in that it won't lock up your console.

```
nginx -c /www/nginx/nginx.conf
```

To stop, run this command.

```
nginx -c /www/nginx/nginx.conf -s stop
```

1.10.5 Reverse Proxy

The below shows how to have Nginx act as a reverse proxy to a Node.js web server listening on port 49000. It also redirects port 80 traffic to the secure port 443 (https) which in turn necessitates SSL configuration.

```
pid /www/mydomain/nginx.pid;
events {}
http {
  server {
    listen 80;
```

(continues on next page)

(continued from previous page)

```
    server_name mydomain.com;
    return 301 https://$server_name$request_uri;
  }
  upstream node_servers {
    server 127.0.0.1:49000;
  }
  server {
    listen 443 ssl;
    ssl on;
    ssl_certificate /www/mydomain/mydomain.com.cert;
    ssl_certificate_key /www/mydomain/mydomain.com.key;
    ssl_protocols TLSv1.2;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA256
    ssl_session_cache shared:SSL:50m;
    ssl_prefer_server_ciphers on;
    location / {
      proxy_pass http://node_servers;
    }
  }
}
```

1.11 PostgreSQL Installation and Basic Configuration on IBM i

The purpose of this document is to summarize PostgreSQL installation and basic configuration on IBM i. The following topics are outlined below: initial setup, enabling remote connections, server startup, and connecting with a client.

1.11.1 Initital Setup

Install the `postgres12-server` and `postgresql-contrib` rpm packages using the IBM i ACS Open Source Package Management wizard or an SSH shell session with:

```
yum install postgresql12-server postgresql12-contrib
```

According to the PostgreSQL docs, "It is advisable to run PostgreSQL under a separate user account. This user account should only own the data that is managed by the server, and should not be shared with other daemons. The user name postgres is often used but you can use another name if you like."

From a 5250 session create a ***POSTGRES*** user profile with:

```
CRTUSRPRF  USRPRF(POSTGRES)  PASSWORD(...)
```

NOTE: Creating a user profile requires `*ALLOBJ` authority.

Next create the `postgres` home directory and change ownership to the ***POSTGRES*** user. From SSH or QSH session run:

```
mkdir /home/postgres
chown postgres /home/postgres
```

NOTE: Creating a directory under `/` requires `*ALLOBJ` authority.

Log in to your IBM i via SSH as the **POSTGRES** user.

If not started, start the bash shell by typing ***bash*** unless bash is already your default shell.

Run the following commands to initialize PostgreSQL database in the `/home/postgres` IFS directory location

```
export PGDATA=/home/postgres

initdb -E UTF-8 -D /home/postgres -W -A scram-sha-256
```

You will be prompted to `Enter new superuser password` for the **POSTGRES** user.

NOTE: This password is for the database and distinct from the **POSTGRES** user profile.

After running `initdb` you should see the following message:

```
Success. You can now start the database server using:
    pg_ctl -D /home/postgres -l logfile start
```

1.11.2 Server Startup

Run the following command to start PostgreSQL database server.

```
pg_ctl -D /home/postgres -l logfile start
```

You should see the following messages:

```
waiting for server to start.... done
server started
```

The following command can be used to stop the server.

```
pg_ctl -D /home/postgres -l logfile stop
```

From a 5250 session, run WRKACTJOB and you should see the active server jobs and threads in the QUSRWRK subsystem

```
------------------------------------------------------------------------------
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
QP0ZSPWP     POSTGRES    BCI       .0  PGM-postgres     SELW
------------------------------------------------------------------------------
```

From a 5250 session, run `NETSTAT *CNN` to verify the server is listening on port 5432. You should see an entry for Local Port 5432 which tells you the server is listening for connections.

```
------------------------------------------------------------------------------
                        Work with IPv4 Connection Status
                                                              System:   SYS1
 Type options, press Enter.
   3=Enable debug   4=End   5=Display details   6=Disable debug
   8=Display jobs

      Remote           Remote     Local
 Opt  Address          Port       Port       Idle Time  State
```

(continues on next page)

(continued from previous page)

```
      *                *          5432       000:05:22  Listen
-------------------------------------------------------------------------------
```

From shell command line, create example pdatabase with the following command:

1.11.3 Connect Client To the Server

The `psql` command line client is a frontend to interact with PostgreSQL server backend. Lets use `psql` to connect to the server, create a database, create a table, insert data, and view the data.

```
$ psql
Password for user postgres:
postgres=# CREATE DATABASE us_states;
CREATE DATABASE

postgres=# \c us_states;
You are now connected to database "us_states" as user "postgres".

us_states=# CREATE TABLE States(id SERIAL, name varchar(50));
CREATE TABLE

us_states=# INSERT INTO States(name) VALUES('Alabama');
INSERT 0 1

us_states=# SELECT * FROM States;
 id |  name
----+---------
  1 | Alabama
(1 row)

us_states=# \q
```

If you enabled remote connections, the same can be done using `psql` from a remote machine with:

```
$ psql -h myhost.example.com -U postgres -d us_states
Password for user postgres:

us_states=# INSERT INTO States(name) VALUES('Alaska');
INSERT 0 1
us_states=# SELECT * FROM States;
 id |  name
----+---------
  1 | Alabama
  2 | Alaska
(2 rows)

us_states=# \q
```

Alternatively you can use a GUI client like pgAdmin, Heidi, DBeaver, etc to connect to Postgres server instead of `psql`.

```
Host: IBMi host name or IP
Port: 5432
User: postgres
```

(continues on next page)

(continued from previous page)

```
Password: Your password
Database: us_states
```

1.11.4 Enable Remote Connections

By default PostgreSQL only listens for client connections from localhost. To allow remote connection we need to configure some files.

NOTE: Secure connections with SSL if your server is accessible publicly. In this example the server protected behind a firewall.

Before editing the configuration files make sure the PostgreSQL server is stopped with

```
pg_ctl -D /home/postgres -l logfile stop
```

Use nano, vim or some other editor to edit ***/home/postgres/postgresql.conf*** file so the server will listen on TCP/IP addresses. We will enable access on all IP addresses

```
# listen_addresses = 'localhost'
listen_addresses = '*'
```

NOTE: This will allow PostgreSQL server to listen for all IP addresses.

Read the docs for more details on Connection configuration.

Edit the `IPv4 local connections` line in ***/home/postgres/pg_hba.conf***

```
# IPv4 local connections:
# TYPE    DATABASE        USER            ADDRESS                 METHOD
# host    all             all             127.0.0.1/32            scram-sha-256
  host    all             all             0.0.0.0/0               scram-sha-256
```

Read the docs for more details on ***pg_hba.conf*** configuration.

NOTE: This will allow clients from any IPv4 address to authenticate.

Start the PostgreSQL server with:

```
pg_ctl -D /home/postgres -l logfile stop
```

Now that you have postgresql installed and setup refer to to the standard PostgreSQL documentation as needed.

1.12 LetsEncrypt w/Certbot

Certbot can be used to get/renew LetsEncrypt certificates. Follow these instructions to install and use Certbot. Certbot's web site can be found at https://certbot.eff.org.

1.12.1 1. SSH into the server

SSH into the server running your HTTP website as a user with *ALLOBJ special authority.

1.12.2 2. Set up your environment

It is assumed that you are running these commands from an SSH terminal.

You must first assure that your PATH environment variable is set up correctly.

```
    PATH=/QOpenSys/pkgs/bin:$PATH
    export PATH
```

1.12.3 3. Install system dependencies

Install Python 3.9 and necessary packages by running:

```
yum install python39-pip python39-cryptography
```

1.12.4 4. Set up a Python virtual environment

Execute the following instruction on the command line to set up a virtual environment.

```
python3.9 -m venv --system-site-packages /opt/certbot
```

1.12.5 5. Install Certbot

Run this command on the command line on the machine to install Certbot.

```
/opt/certbot/bin/pip install certbot
```

1.12.6 6. Choose how you'd like to run Certbot

Are you ok with temporarily stopping your website?

Yes, my web server is not currently running on this machine.

Stop your webserver, then run this command to get a certificate. Certbot will temporarily spin up a webserver on your machine.

```
/opt/certbot/bin/certbot certonly --standalone
```

No, I need to keep my web server running.

If you have a webserver that's already using port 80 and don't want to stop it while Certbot runs, run this command and follow the instructions in the terminal.

```
/opt/certbot/bin/certbot certonly --webroot
```

Important Note:

To use the webroot plugin, your server must be configured to serve files from hidden directories. If /.well-known is treated specially by your webserver configuration, you might need to modify the configuration to ensure that files inside /.well-known/acme-challenge are served by the webserver.

1.12.7 7. Install your certificate

Proper technique will vary depending on the web server in use. If using IBM i system Apache, the DCM Tools project may be useful.

1.12.8 8. Confirm that Certbot worked

To confirm that your site is set up properly, visit https://yourwebsite.com/ in your browser and look for the lock icon in the URL bar. Most browsers will also let you inspect the certificate by clicking on the lock icon.

1.12.9 9. Renewing your certificate manually

Certificate renewal can be done by running the following in your terminal:

```
/opt/certbot/bin/certbot renew
```

You can do a "dry run" of the renewal (making no modifications) by running:

/opt/certbot/bin/certbot renew –dry-run

1.12.10 10. Setting up automatic renewal

To set up automatic renewal of your certificate, first create a shell script that performs the following tasks:

- Stops your web server
- Runs /opt/certbot/bin/certbot renew -q
- Installs the certificate
- Starts your web server
- (optional) Records the activity, for instance by running `system "SNDMSG MSG('Certificate renewal process complete') TOUSR(*SYSOPR)"`

For instance, the following script uses Service Commander to restart the app. It also writes output to `renewcert.log` and sends a message when completed.

```
#!/QOpenSys/pkgs/bin/bash
export PATH=/QOpenSys/pkgs/bin:$PATH
set -e
cd $(dirname $0)
exec >> renewcert.log
exec 2>&1
echo "================================================="
date
sc stop mywebapp
/opt/certbot/bin/certbot renew
sc start mywebapp
system "SNDMSG MSG('Certificate renewal process has completed') TOUSR($(/usr/bin/id -
↪u -n))"
```

Once that is completed, you can create a job scheduler entry that calls your script. This example Shows how to create a job scheduler entry that runs at 1:11 AM on the first and third sundays.

```
ADDJOBSCDE JOB(CERTRENEW) CMD(QSH CMD('/path/to/script.sh')) FRQ(*MONTHLY)␣
↪SCDDATE(*NONE) SCDDAY(*SUN) SCDTIME(011111) RELDAYMON(1 3) SAVE(*YES)
```

Replace the job name and path to script as needed.

1.12.11 11. Upgrading Certbot

It's good practice to occasionally update Certbot to keep it up-to-date. To do this, run the following command on the command line on the machine.

```
/opt/certbot/bin/pip install --upgrade certbot
```

If this step leads to errors, run `rm -rf /opt/certbot` and repeat all installation instructions.

1.12.12 What about wildcard certificates?

If you need wildcard certificates, follow steps 1-5, above, then proceed as documented here

6. Check if your DNS provider is supported

See if your DNS provider is supported by Certbot by checking this list in the documentation.

Not supported?

If your DNS provider is not supported, pause here: run Certbot with the manual plugin by using these steps from the documentation.

Supported?

If your DNS provider is supported, continue with the remaining instructions below in your SSH terminal.

7. Install correct DNS plugin

Run the following command, replacing with the name of your DNS provider.

```
/opt/certbot/bin/pip install certbot-dns-<PLUGIN>
```

For example, if your DNS provider is Cloudflare, you'd run the following command:

```
/opt/certbot/bin/pip install certbot-dns-cloudflare
```

8. Set up credentials

You'll need to set up DNS credentials. Follow the steps in the "Credentials" section for your DNS provider to access or create the appropriate credential configuration file. Find credentials instructions for your DNS provider by clicking the DNS plugin's name on the Documentation list.

9. Get a certificate

Run one of the commands in the "Examples" section of the instructions for your DNS provider.

10. Install and renew certificate

Follow steps 7 and beyond, above.

1.13 ACS Clone Repo Tool

The "Clone Repo" tool is a tool that allows you to clone any http or https-hosted RPM repository to the local Integrated File System (IFS) of the target IBM i system.

1.13.1 How do I launch the "Clone Repo" tool?

1. In Access Client Solutions, first access the Open Source Package Management tool (Tools->"Open Source Package Management")
2. The clone tool is available after signing onto a system with the open source package management tool (Utilities->"Clone Repo for Offline Use")

1.13.2 Interface options, explained

Source Repository

Specify the original repository that you'd like to take a clone/snapshot of.

Destination (IFS)

Specify a target directory on IBM i for the clone/snapshot.

Additional Operations -> Create or Update Repository Definition

The YUM package manager only knows about repos that are defined in YUM's repository list. The repository list is simply a set of `.repo` files in the `/QOpenSys/etc/yum/repos.d/`

Additional Operations -> Disable Repositories that Require Internet Access from the IBM i System

By default, YUM will fail any operations if it can't read from all the configured repositories. This options disables Internet-requiring repos, so that YUM operations continue to work. Keep this option checked if your IBM i system can't access the Internet.

Additional Operations -> Create nginx configuration file

Creates a configuration for the nginx http server to allow you to host this repo clone via http, so that other systems in your network can access it (more details below).

1.13.3 Serving up your internal repo via nginx using ACS

The ACS "Clone Repo" tool makes it easy to serve your cloned repo to other systems in your network. If you check the "create nginx configuration file" option, the following files are created for you:

- nginx.conf: Used by nginx. By default, it configures nginx to use 5 worker processes and listen on port 2055. Feel free to customize to suit your needs.
- startServer: Starts nginx as a background task in current subsystem.
- startServerBatch: Submits nginx to QHTTPSVR subsystem (feel free to customize to suit your needs).
- stopServer: Stops the nginx instance

Of course, the next step is to configure other systems to point at your newly-created repo. That can be done on endpoint systems either by:

1. installing `yum-utils` and invoking `yum-config-manager --add-repo <ip-address-where-hosted>` (for instance, `yum-config-manager --add-repo http://mysystem:2055`).
2. creating a repository definition in `/QOpenSys/etc/yum/repos.d`. A repository definition is a small text file with basic information. Use `ibm.repo` as an example.

To automate yum updates, one can use a job scheduler entry, specifying `-y` so the YUM tool doesn't stop to ask for user confirmation. Example: `ADDJOBSCDE JOB(YUMUP) CMD(QSH CMD('exec /QOpenSys/pkgs/bin/yum -y upgrade')) FRQ(*WEEKLY) SCDDAY(*ALL)` This example does a daily upgrade, but note that no update

will happen if the configured repo or repos have no changes. If the only configured repo is your private one, then this will not do anything until you update your repo.

To summarize, the process of creating your own repository and hosting it for all your systems involves:

1. Run the ACS Clone Repo Tool, checking the "Create nginx configuration file" box
2. Start the nginx server by running the `startServer`/`startServerBatch` script
3. Configure endpoint systems
4. (optional) automate

(A good practice might to to have a different repo for each class of systems, such as development, test, and production)

1.14 IBM Repositories

IBM provides a set of repositories to use with yum. Currently, this is one repo called "ibm" pointing here. This repo file is setup and enabled during the bootstrap installation, however admins are free to disable or change this file at their discretion.

A change has been pushed out to yum and the bootstrap to install the "ibmi-repos" package which provides two new repos:

- ibmi-base
- ibmi-release

The ibmi-base repo will point to the same location as ibm (at least for now), making the ibm repo redundant and can be removed manually.

The ibmi-release is a new repo that points to a release-specific directory. Depending on what IBM i version you are running, the repo will dynamically determine the correct path to point to. This repo will contain rpms which are built specifically for each IBM i release.

1.14.1 Transition

Currently, the ibm repo is provided by the bootstrap, but is not owned by a package. Because admins may have modified this file and it's not tracked by rpm, the decision was made to replace this file with a new repo file in the rpm. This means any modifications will not be overwritten by the new rpm, but it does mean that you may have a superfluous repo file that has to be cleaned up manually.

1.14.2 FAQs

Why are you making this change?

By providing the IBM repos in an rpm, it allows us to push out updates to our repo files using the same yum update mechanism.

Some examples of changes may include:

- The path to repo needs to change
- The protocol changes (FTP -> HTTPS)
- We start signing our repo metadata
- etc

Why are you adding a new ibm-base repo instead of updating the existing repo?

Because ibm.repo is not tracked by rpm, it becomes tricky to take over ownership of the file by rpm while preventing any local modifications from being overwritten while also being a seamless transition for users. By providing a new repo file, all users will immediately get access to the new repo because there are no conflicts with existing repo files. This also allowed us to give the repo a more meaningful name.

What if I've deleted or disabled the ibm repo file?

Because the ibmi-repos package ships new repo files, the existing repo file is unaffected. However, the new repos *will* be enabled by default. You can disable them using `yum-config-manager --disable ibmi-base ibmi-release`

Note that the new repo files are shipped with `skip_if_unavailable=1` set. This means that if you've disabled the ibm repo because your system does not have internet access, these repo files will not cause yum failures, but only warnings like

```
https://public.dhe.ibm.com/software/ibmi/products/pase/rpms/repo-7.4/repodata/repomd.
↪xml: [Errno 14] HTTPS Error 404 - Not Found
Trying other mirror.
```

What if I've made changes to the existing ibm repo file?

Because the ibmi-repos package ships new repo files, the existing repo file is unaffected. It is recommended to rename the ibm.repo and disable the new repos with `yum-config-manager --disable ibmi-base ibmi-release` or manually apply your changes to the new repo files.

What happens if I make changes to the new ibmi-base or ibmi-release repos?

Both of these files are marked as special config files inside rpm, which means that rpm will detect if they've been modified and if so it **will not** overwrite them. Instead, you will see a message from yum:

```
warning: /QOpenSys/etc/yum/repos.d/ibmi-base.repo created as /QOpenSys/etc/yum/repos.
↪d/ibmi-base.repo.rpmnew
```

It is up to you to look at the .rpmnew file to see what changes have been made and, if applicable, adjust the repo file accordingly.

You said ibm and ibmi-base point to the same location; will that cause any problems?

There should not be any problems with both ibm and ibmi-base pointing to the same location. Yum will see two repos providing the same packages and will pick one when it goes to install updates.

Though there should be no problems having them both enabled, it's recommended that you disable the ibm repo using `yum-config-manager --disable ibm`. Once you have verified yum continues to work without it enabled, you can remove it using `rm /QOpenSys/etc/yum/repos.d/ibm.repo`.

What should I do with the existing ibm repo file?

This repo is no longer needed as the ibmi-base repo supersedes it. It's recommended that you disable the ibm repo using `yum-config-manager --disable ibm`. Once you have verified yum continues to work without it enabled, you can remove it using `rm /QOpenSys/etc/yum/repos.d/ibm.repo`.

Can I remove the ibmi-repos package?

No, yum has been made to depend on it and yum will prevent removing itself.

If you do not want these new repo files to be enabled, you can disable them using `yum-config-manager --disable ibmi-base ibmi-release`.

Will the ACS clone tool work with the new repos?

Yes, the ACS clone tool should work with any repository.

Will the ACS yum proxy work with the new repo?

Yes, the ACS yum proxy should work with all HTTP-based repositories.

1.15 Third-party (non-IBM) repositories

The repositories listed on this page are not owned, managed, or supported by IBM. However, the repositories have been inspected and the software generally seems to be built with IBM-approved conventions for existing well in the IBM-delivered open source ecosystem.

1.15.1 Installation instructions

First the `yum-utils` package should be installed. This provides the `yum-config-manager` utility, which mades it easy to add new repositories, as well as enable or disable existing repositories.

1.15.2 Repository List

PHP

Several PHP distributions are available. See http://ibm.biz/ibmi-php

OpenMax

Brought to you by: Massimo Fantin

Software offered: A large collection of open source packages, including Postgresql, Ghostscript, Postfix , rddtool etc etc

The i Doctor

Brought to you by: Jack Woehr

Software offered: lynx-dev (limited capabilities, for instance no https support). schily-tools (cdrecord, mkisofs, etc.)

Install: `yum-config-manager --add-repo http://www.the-i-doctor.com/oss/repo/the-i-doctor.repo`

QSECOFR

Brought to you by: Yvan Janssens

Software offered: Mono on i and various Open Source software for which fixes have been accepted upstream to enable them to run on i.

Install: `yum-config-manager --add-repo https://repo.qseco.fr/qsecofr.repo`

SoBored

Brought to you by: Josh Hall

Software offered: ibmi-dotfiles

Install: `yum-config-manager --add-repo http://rpms.sobo.red/ibmi/`

1.16 Porting Software to PASE

1.16.1 Contents

Gotchas When Building Software in PASE

malloc behaves differently from many other platforms

Zero byte allocations on AIX can be strange. If software depends on this behaviour, you can build with `-D_LINUX_SOURCE_COMPAT` to enable a built-into-libc wrapper that has the glibc behaviour.

C++ issues

Make sure that you have `libstdcplusplus-devel` in addition to g++ installed.

If you run into issues with the threading parts of the C++ standard library, such as `std::mutex`, ensure that `-pthread` is passed to g++. This will flip some defines and such so that threading is exposed. Linking pthread (`-pthread` if you use GCC as the linker, otherwise `-lpthread`) in also helps with crashes in C++ standard library locale code.

AF_LOCAL not declared

```
error: 'AF_LOCAL' was not declared in this scope
     c = socket(AF_LOCAL, SOCK_STREAM, 0);
                ^~~~~~~~
```

On some platforms, `AF_LOCAL` is provided as a synonym of `AF_UNIX`, but not on AIX/PASE. Either alter the code to use `AF_UNIX` directly or add your own synonym:

```
#ifndef AF_LOCAL
#define AF_LOCAL AF_UNIX
#endif

// ...
c = socket(AF_LOCAL, SOCK_STREAM, 0);
```

TOC overflow

```
ld: 0711-781 ERROR: TOC overflow. TOC size: 85080       Maximum size: 65536
collect2: error: ld returned 12 exit status
```

Understanding the TOC (table of contents) could take up its own entire article. Indeed there's a good one here. The usual way to fix is by passing `-bbigtoc` option to the linker (`ld`). Using GCC, you need to prefix linker flags with `-Wl,` and these flags are specified in `$LDFLAGS`:

```
export LDFLAGS='-Wl,-bbigtoc'
./configure ...
make ...
```

While `-bbigtoc` should always do the job, it's not the most efficient. GCC also has additional ways to reduce TOC pressure that may be better performing, but you'll need to understand the code you're trying to compile and what each option does to apply them. Look for more info on the following compiler options here:

- `-mno-fp-in-toc`
- `-mno-sum-in-toc`
- `-mminimal-toc`
- `-mcmodel=medium`
- `-mcmodel=large`

`readdir_r` return values

On AIX and PASE, `readdir_r` returns 9 and sets the result to NULL for both end of directory and error; on error, it sets `errno`. This is unlike other Unix systens and very easy to use incorrectly even when adapting to its quirks, so you can try:

- Set `errno` to 0 before calling. This means you can disambiguate between the cases.
- Build with `-D_LINUX_SOURCE_COMPAT`. This will point `readdir_r` to an alternative version with glibc-like semantics.

`uname` return values

Mostly a cosmetic issue, but applications expecting a full version in the release field of utsname will be disappointed. AIX and PASE put the major version (you know, the "V" in "V7R3") in version, and the minor version (the "R") in release.

Thread safety

AIX and PASE are **not** thread safe by default. You should pass `-D_THREAD_SAFE`, which enables important things like a thread-safe `errno`.

VFS woes

`mntctl`, `statvfs`, and ILE `statvfs` can never agree on a consistent value for magic names and numbers, unlike AIX. Try to avoid converting between their values.

SIOCGIFCONF

On PASE, `SIOCGIFCONF` doesn't show the true line description name, nor does it return an `AF_LINK` entry; the device will have a fake name based on its IPv4 address. This makes getting the interface index and MAC address nearly impossible; you can't even use `if_nameindex` because that returns the true line description name. As an alternative, you can use `Qp2getifaddrs`, which works like most Unix-like systems' `getifaddrs`.

Map PASE/IBM i Version to AIX Version

IBM i Version	AIX Version
7.4	7.2 TL2
7.3	7.1 TL4
7.2	7.1 TL1
7.1	6.1 TL2
6.1	5.3 TL6
5.4	5.3 ML3

1.17 Guide to migrating from 5733-OPS to RPMs

1.17.1 Environment setup (PATH)

By default, the RPM-form packages do not create symbolic links in standard, used-by-default directories like `/QOpenSys/usr/bin/` or `/usr/bin/`.

To address this, Set the PATH environment variable so that your shell will find the new commands when you try to run them.

1.17.2 Modifying scripts to use an appropriate "shebang" (#!) line

When writing a shell script (or a Python/Node.js program), it is common practice to start your source code with a "shebang" line(`#!`). This tells the shell what program to use when the script is run.

To properly use RPM-form executables, there are two options for making sure the scripts use the RPM form, and not 5733-OPS.

The first (and most industry-standard) technique is to use the special `/usr/bin/env` command in the shebang line, followed by the program that is interpreting the script. This will look in your PATH for the named executable. Thus, it is very important that your PATH be set correctly!!! For instance, for a bash script:

```
#!/usr/bin/env bash
echo "starting my shell script"
```

Another approach is to fully-qualify the path to the binary in `/QOpenSys/pkgs/bin`. For instance:

```
#!/QOpenSys/pkgs/bin/bash
echo "starting my shell script"
```

These techniques can apply to more than just shell scripts, and can be used for other interpreted languages such as perl or Python. For example, here's how the shebang line would look for a `python3.9` program with the `/usr/bin/env` method

```
#!/usr/bin/env python3.9
print("starting my python program")
```

And with the fully-qualified technique:

```
#!/QOpenSys/pkgs/bin/python3.9
print("starting my python program")
```

1.17.3 Python migration notes

See *Python usage notes*

1.17.4 Node.js migration notes

See *Node.js usage notes*

CHAPTER

TWO

COLLABORATION

- Questions/comments about this site.
- Ryver forum (join here)

CHAPTER

THREE

WHERE TO FIND EXAMPLES

- IBM i OSS Examples (GitHub)

CHAPTER

FOUR

ACCESS TO SOURCE CODE

- Some open source licenses, such as GPL, require that source code necessary for building the software (including patches, build recipes, etc) be made available to the public. For these packages, the source is provided in the form of source RPMs distributed at http://public.dhe.ibm.com/software/ibmi/products/pase/rpms/repo/src/

CHAPTER

FIVE

LINKS

- Previous authoritative IBMiOSS site