



APPX LIBRARY

PHP ChartDirector for APPX

6.2.1

Rocky Linux 9 and 10 and Windows Server 2025; 32-bit and 64-bit APPX engines

October 2026. Rocky AppStream PHP 8.3 on Linux; official PHP 8.5 on Windows. Step-by-step APPX desktop walkthroughs and SELinux guidance.

APPX Software, Inc.

Prepared October 16, 2026. PHP CLI installation and APPX chart use on Rocky Linux and Windows Server. Examples are embedded in this PDF.

Contents

Chapter 1 Plan the Installation.....	4
Start Here	4
Versions and Architecture.....	4
Names and Locations	5
Installation Order	5
Chapter 2 Install on Rocky Linux	7
Choose the Linux Route	7
Install Natively on Rocky Linux 9.....	8
Prepare the 32 Bit Compatibility Libraries	8
Install PHP CLI	9
Install ChartDirector	10
Supply the APPX libtar Dependency.....	11
Load the Extension for CLI.....	11
Prepare Output and the APPX Environment	12
SELinux Allowances for Chart Rendering	15
Chapter 3 Install on Windows Server.....	17
Install the Current PHP CLI	17
Install the Microsoft Runtime	17
Install and Configure ChartDirector.....	18
Set Permissions and Service PATH	19
Chapter 4 Verify PHP Before APPX.....	20
Render the Smoke Test.....	20
Test as the APPX Runtime Account.....	22
Purchase and License ChartDirector.....	23
Optional Browser Demonstrations	23
Chapter 5 Connect APPX to ChartDirector	24
Prepare the APPX Installation	24
Set the Chart Parameters.....	28
Run a 1CD Example	34
Capture a Render Failure.....	37
Chapter 6 Create and Display a Chart	38
Build a Small APPX Chart	38
Render and Check the Returned Path	38

Display the Image in an Input.....	39
Include the Chart in a PDF Report	39
Chapter 7 Operate and Maintain the Installation	40
Prevent Stale Charts and Filename Collisions.....	40
Clean Up Generated Files	40
Upgrade PHP or ChartDirector.....	40
Chapter 8 Troubleshoot by Symptom.....	41
PHP Cannot Be Found.....	41
The Extension Does Not Load.....	41
APPX Reports Error Creating Image.....	41
The PNG Is Missing or the Wrong Image Appears.....	42
Fonts or External Images Are Missing.....	42
A 32 Bit Engine Does Not Start on Rocky Linux 10.....	42
Appendix A Reference and Checks	43
Installation Checklist	43
Downloads and Sources	43
Use the embedded examples	44

CHAPTER 1

Plan the Installation

Choose the platform and architecture before installing.

Start Here

This guide installs PHP ChartDirector for APPX 6.2.1 on Rocky Linux 9, Rocky Linux 10 or Windows Server 2025. It covers the PHP configuration, the APPX chart parameters, and a first chart through the APPX desktop client. The later chapters explain how to use a chart in an application screen or PDF report.

APPX starts PHP as a separate command line program. Apache, IIS, PHP-FPM, a web root and an HTTP port are not required for this procedure. Keep ChartDirector and chart output outside any public web directory. An existing PHP website can use a different PHP installation and configuration.

Choose the PHP edition. Download ChartDirector for PHP. The newer version numbers offered for ChartDirector C++ and .NET are different products and do not identify the PHP release.

APPX 6.2.1 provides 32-bit and 64-bit engines.

Versions and Architecture

The examples use Rocky AppStream PHP 8.3 on Linux, official PHP 8.5 x64 Non Thread Safe on Windows, and ChartDirector for PHP. The package tested in October 2026 is labelled ChartDirector 7.0 (PHP Edition). Use the current maintenance release of the selected PHP branch. Your installed PHP version may differ; select its matching ChartDirector extension from the vendor's [installation table](#).

Linux packages come from Rocky BaseOS, AppStream, Extras and CRB. Matching EPEL packages may be used for prerequisites. No third-party PHP repository is needed. APPX remains at 6.2.1 in this edition.

Operating system	APPX engine	Route
Rocky Linux 9 x86_64	6.2.1 x86 or x64	Native host; match dependency architecture
Rocky Linux 10 x86_64	6.2.1 x64	Native architecture; verify listed runtime dependencies
Rocky Linux 10 x86_64	6.2.1 x86	Separate compatibility userspace required
Windows Server 2025 x64	6.2.1 x86	x86 APPX engine with x64 PHP CLI
Windows Server 2025 x64	6.2.1 x64	x64 APPX engine with x64 PHP CLI

The vendor's PHP package label and the version returned by its native library can differ. The tested package is labelled 7.0 and its library reports 7.1.0. Select the extension by the PHP version, architecture and thread-safety setting below.

Match the **ChartDirector extension to PHP**, including operating system, PHP minor version, PHP architecture and thread safety. APPX and PHP run in separate processes; their bitness need not match. A 32-bit APPX engine can therefore invoke 64-bit PHP and its 64-bit ChartDirector extension on 64-bit Windows or within a prepared Rocky 9 environment. That does not let a 32-bit PHP process load a 64-bit extension.

PHP used by these recipes	ChartDirector extension
PHP 8.3 x64 NTS on Linux	phpchartdir830.dll from the Linux x86_64 package
PHP 8.5 x64 NTS on Windows	phpchartdir850nts.dll from the Windows x64 package
PHP 8.5 x64 TS on Windows, if deliberately selected	phpchartdir850.dll from the Windows x64 package

The `.dll` suffix is also used by the Linux PHP extension. Do not rename it to `.so`. `libchartdir.so` is its dependency, not a second PHP extension to add to `php.ini`.

Names and Locations

The following paths are examples. Your installation directories, APPX account and service name will probably differ. Substitute your own values in commands and APPX fields, and use the same choices throughout the installation. The examples call the runtime account `appx` and its Linux group `appxgrp`; check the account actually running your APPX sessions.

Component	Rocky Linux	Windows Server
PHP CLI	<code>/usr/bin/php</code>	<code>C:\AppxTools\PHP85\php.exe</code>
ChartDirector library	<code>/opt/ChartDirector/lib</code>	<code>C:\AppxTools\ChartDirector\lib</code>
Shared PHP interface	<code>/opt/ChartDirector/lib/ phpchartdir.php</code>	<code>C: \AppxTools\ChartDirector\lib\phpchartdir. php</code>
Test and example scripts	<code>/opt/appx-chart-examples</code>	<code>C:\AppxTools\ChartExamples</code>
Generated chart files	<code>/var/lib/appx-charts</code>	<code>C:\AppxData\Charts</code>

Set the OCD PHP Library parameter to the full `phpchartdir.php` filename. Use a separate writable directory for chart output. Both are server filesystem paths.

The example chart size, 640 by 400 pixels, is a starting point. Choose dimensions appropriate to your application's screen or printed report. The shipped 1CD samples can set their own sizes.

Installation Order

1. Back up APPX and choose the engine, platform and installation route.
2. Install PHP CLI and ChartDirector, then render as the APPX account.

3. Configure 0CD and run the 1CD and application examples.
4. Check the screen, saved report, refresh and concurrent sessions.

CHAPTER 2

Install on Rocky Linux

Install and check the Linux command line runtime.

Choose the Linux Route

Rocky Linux 10 uses an x86-64-v3 baseline and no longer supplies the i686 compatibility packages used by 32-bit applications. Use an x86_64 machine with a compatible CPU. An ARM instance does not run the Intel APPX engine natively.

Apply the current Rocky updates before installing. If an update installs a new kernel, reboot during the server's maintenance window. A container uses its host's kernel.

The 64-bit APPX 6.2.1 engine runs natively on Rocky Linux 10. Its 32-bit engine requires the following compatibility route. On Rocky Linux 9, either engine can use a native installation.

Rocky Linux 10 cannot run this 32-bit APPX engine with its standard packages. Use a complete 32-bit-capable Rocky Linux 9 userspace in a container on the Rocky 10 host, or put the APPX engine in a supported separate virtual machine. Install current Rocky AppStream x64 PHP 8.3 and ChartDirector x64 inside that same userspace. A host PHP installation is not automatically available inside a container.

For a compatibility evaluation on the Rocky 10 host, install Podman, then create a dedicated container. The host paths below must be dedicated to this test; the `:Z` option gives them a private container label.

```
sudo dnf install -y podman
sudo mkdir -p /srv/appx32-chartlab /srv/appx32-chartlab-charts
sudo podman run --name appx32-chartlab -it \
  --hostname appx32-chartlab \
  -v /srv/appx32-chartlab:/usr/local/appx:Z \
  -v /srv/appx32-chartlab-charts:/var/lib/appx-charts:Z \
  docker.io/library/rockylinux:9 /bin/bash
```

Inside that container, install its current Rocky 9 updates, its Rocky repositories, CRB and EPEL, the required i686 runtime libraries, x64 PHP 8.3 and ChartDirector x64. Run the APPX installer inside the container too. All PHP, ChartDirector and output paths in this guide must refer to locations visible inside the container. Select the Rocky 9 AppStream php:8.3 stream inside the userspace.

Keep the container and its APPX data separate from other installations. An interactive evaluation container does not automatically provide persistent APPX services or desktop-client networking. Establish those deployment requirements independently before using this route for a server. For an installation requiring ordinary service management, use a separate supported guest operating system. This compatibility userspace is an explicit exception to an all-Rocky-10 runtime stack.

Do not copy a collection of Rocky 9 shared libraries into Rocky 10's `/usr/lib` or `/usr/lib64`. Keep a compatibility userspace together in its own container or virtual machine. The PHP process and ChartDirector library must be accessible inside the same execution environment that launches the chart.

Install Natively on Rocky Linux 9

Rocky Linux 9.8 provides a native route for APPX 5.5.4 x86 and both APPX 6.2.1 engine builds. This avoids a compatibility container when you can choose the host operating system. Use the latest Rocky 9 updates and reboot into the installed kernel before testing.

For a 32-bit APPX engine, follow the i686 package and libtar build recipe in the next topic directly on the Rocky 9 host. For APPX 6.2.1 x64, use the native libtar rebuild topic, changing the local RPM suffix from `.el10.local` to `.el9.local`. Keep the x86_64 and i686 packages matched to their respective engines.

Use Rocky 9 AppStream's **php:8.3** stream and its current x64 `php-cli` package for every APPX engine choice. The executable and ini paths are the same on Rocky 9 and 10. Keep each APPX release/build in a separate directory, complete normal APPX registration, and configure its own OCD parameters.

The native Rocky 9 tests rendered the shipped `1CD B01. SIMPLE BAR CHART` through APPX 5.5.4 x86, 6.2.1 x86 and 6.2.1 x64. The architecture and PHP invocation are independent: native 32-bit APPX invoked the same x64 PHP CLI successfully.

Prepare the 32 Bit Compatibility Libraries

Use this topic only for the 32-bit engine route. Run these commands **on a native Rocky 9 host or inside the Rocky 9 container or guest** selected above. The tested userspace was Rocky 9.8. Keep its updated x64 libraries and i686 packages within that Rocky 9 environment.

```
sudo dnf upgrade --refresh -y
sudo dnf install -y glibc.i686 libstdc++.i686 krb5-libs.i686 \
    libxml2.i686 xz-libs.i686 zlib.i686 libxcrypt-compat.i686 \
    rpm-build dnf-plugins-core gcc make glibc-devel.i686 pam.i686
```

The APPX 5.5.4 engine additionally needs 32-bit libxml2; both tested 32-bit engines need 32-bit Kerberos and libtar. The 6.2.1 engine needs 32-bit liblzma from `xz-libs.i686`. Install the explicit packages above even if a PHP or compiler dependency has already installed their x64 counterparts.

The tested Rocky 9.8 repositories did not supply libtar either. The separate compatibility procedure rebuilds Rocky's own libtar source for i686 inside this userspace. It produces a local RPM for Rocky 9; it is not an official Rocky 9 repository package. No Rocky 8 binary repository is enabled:

```

mkdir -p "$HOME/libtar32-build"
cd "$HOME/libtar32-build"
curl -fL -o libtar.src.rpm \
    https://dl.rockylinux.org/pub/rocky/8/Devel/source/tree/Packages/l/
libtar-1.2.20-17.el8.src.rpm
printf '%s %s\n' \
    97828b3e431caa5f180d7153bbe14a90e6f049eedf0619b63a56cfcb8ff01387 \
    libtar.src.rpm | sha256sum -c -
sudo dnf builddep -y libtar.src.rpm
rpmbuild --rebuild --target i686 --define "_topdir $PWD" \
    --define 'dist .el9.local' \
    --define 'optflags -O2 -g -m32 -march=i686 -mtune=generic -fPIC' \
    --define '__global_ldflags -m32' libtar.src.rpm
sudo dnf install -y RPMS/i686/libtar-1*.rpm

```

Run the build as an ordinary build user. Use `sudo` for package installation; a root shell inside the dedicated container can omit `sudo`. After installing APPX, run `ldd` on the actual engine inside this userspace and resolve every missing library. Both the 5.5.4 and 6.2.1 i686 loader checks passed with this locally built package. Both engines also rendered the shipped native 1CD simple-bar chart inside the Rocky 9.8 container on the updated Rocky 10 host. The reference appendix distinguishes these native tests from application screen and report checks.

Install PHP CLI

Run these commands inside the Rocky environment that launches APPX: the native host or the dedicated Rocky 9 compatibility userspace. Do not configure host PHP for an APPX engine confined to a container.

Use Rocky's current AppStream PHP CLI. On the tested Rocky 9.8 image, the newest available PHP stream was 8.3; Rocky 10.2 supplied PHP 8.3 directly. Both installed PHP 8.3.33 x64 NTS from Rocky's signed packages. Enable only the appropriate Rocky repositories and EPEL for the operating system you are actually running.

```

sudo dnf install -y dnf-plugins-core epel-release
sudo dnf config-manager --set-enabled crb
sudo dnf upgrade --refresh -y

```

On a **fresh Rocky 9** installation, select its AppStream PHP stream before installing the CLI:

```

sudo dnf module list php
sudo dnf module enable -y php:8.3

```

Rocky 10 does not need that module-selection step. If another PHP stream is already enabled on an existing Rocky 9 server, plan and review the stream change before proceeding; do not reset a working PHP installation blindly.

```
sudo dnf install -y php-cli tar gzip file
/usr/bin/php -v
/usr/bin/php --ini
/usr/bin/php -i | \
    grep -E 'Architecture|Thread Safety|PHP API|extension_dir'
rpm -qi php-cli
dnf repolist
```

The normal executable is `/usr/bin/php`, with `/etc/php.ini` and `/etc/php.d/`. Install `php-cli` for this recipe; a web server is optional. Use the newest maintained package in the selected Rocky stream and repeat these checks after updates. Match ChartDirector to the installed PHP minor version; the Linux recipe below uses the PHP 8.3 extension.

Install ChartDirector

1. Download the Linux x86_64 PHP archive from the [ChartDirector downloads](#) page.
2. Record its SHA-256 checksum and retain the original archive with your deployment record. The vendor updates the archive behind a stable download URL; the saved checksum identifies your exact copy.
3. Extract it under `/opt` so the library is `/opt/ChartDirector/lib`.

```
mkdir -p /tmp/chartdirector-download
cd /tmp/chartdirector-download
curl -fL -o chartdir_php_linux_64.tar.gz \
    https://www.advsofteng.com/chartdir_php_linux_64.tar.gz
sha256sum chartdir_php_linux_64.tar.gz
sudo tar -xzf chartdir_php_linux_64.tar.gz -C /opt
ls /opt/ChartDirector/lib/phpchartdir830.dll
file /opt/ChartDirector/lib/phpchartdir830.dll
ldd /opt/ChartDirector/lib/phpchartdir830.dll
```

Install `freetype`, `fontconfig` and `dejavu-sans-fonts` before running the APPX installer on a minimal Rocky image. Its bundled Java runtime uses FreeType even for unattended installation. These prerequisites do not require a web server or desktop environment.

```
sudo dnf install -y freetype fontconfig dejavu-sans-fonts
```

Keep the complete `lib` directory, including `libchartdir.so`, `phpchartdir.php` and `fonts`. Keep its files readable by the APPX runtime identity and its directories traversable. The runtime account needs write access to chart output, not to the vendor library.

Supply the APPX libtar Dependency

On native Rocky 9 or Rocky 10, the APPX 6.2.1 x64 engine requires `libtar.so.1`. The tested Rocky 9.8 and 10.2 repositories used for this cookbook did not supply it. The tested solution rebuilds the Rocky 8 source RPM on the destination Rocky release and installs the resulting local x86_64 RPM. This is a locally maintained compatibility dependency, separate from the official APPX installer and the signed Rocky 9/10 repositories. Rebuilding Rocky source does not make the result an official Rocky 9/10 package. No Rocky 8 binary repository is enabled.

```
mkdir -p "$HOME/libtar-rocky10"
cd "$HOME/libtar-rocky10"
curl -fL -o libtar.src.rpm \
  https://dl.rockylinux.org/pub/rocky/8/Devel/source/tree/Packages/l/
libtar-1.2.20-17.el8.src.rpm
printf '%s %s\n' \
  97828b3e431caa5f180d7153bbe14a90e6f049eedf0619b63a56cfcb8ff01387 \
  libtar.src.rpm | sha256sum -c -
sudo dnf install -y rpm-build dnf-plugins-core gcc make binutils
sudo dnf builddep -y libtar.src.rpm
rpmbuild --rebuild --define "_topdir $PWD" \
  --define 'dist .el10.local' libtar.src.rpm
sudo dnf install -y RPMS/x86_64/libtar-1*.rpm
ldd /path/to/appx
```

On Rocky 9, change `.el10.local` to `.el9.local` throughout the example; keep the build target `x86_64`. Run the build as an ordinary build user and use `sudo` only for package management. Adjust `/path/to/appx` to the actual native 64-bit engine. The accepted local package is `libtar-1.2.20-17.el10.local.x86_64`. Confirm that `ldd` reports no missing libraries, and record your own built RPM checksum. Do not substitute an i686 package for the x64 engine or copy a prebuilt library from a different operating system release.

This native x64 recipe does not resolve the absent 32-bit loader on Rocky 10. Follow the separate compatibility route for 32-bit APPX.

Load the Extension for CLI

Follow the [current ChartDirector PHP installation guide](#): load the matching extension by its absolute filesystem path and keep the complete vendor library directory together. The Linux filename ends in `.dll` even though it is a Linux shared library.

Create `/etc/php.d/40-chartdirector.ini` containing:

```
extension=/opt/ChartDirector/lib/phpchartdir830.dll
```

Then inspect the configuration and module list:

```
/usr/bin/php --ini
/usr/bin/php -m
/usr/bin/php -r \
'printf("PHP %s, %d-bit, %s\n", PHP_VERSION, PHP_INT_SIZE * 8, PHP_ZTS ? "TS" : "NTS");'
```

The vendor also supports loading a bare extension filename from PHP's `extension_dir`. For that alternative, copy **all** of `ChartDirector/lib`, including its `fonts` subdirectory, into that directory. Copying only `phpchartdir830.dll` is insufficient. The absolute-path method above avoids that extra copy.

Every new CLI invocation reads its configuration. There is no PHP web service to restart for a CLI-only change. If the extension fails to load, resolve the startup error before testing APPX. Changing a PHP-FPM pool or Apache configuration does not change this CLI interpreter.

Prepare Output and the APPX Environment

Choose the directories and runtime account

PHP reads the ChartDirector library and example script, then writes a new chart image. The examples keep those three kinds of files in separate directories:

Directory	What belongs there	Access needed by the APPX account
<code>/opt/ChartDirector/lib</code>	The vendor's PHP include files, native extension, companion library and fonts installed earlier in this chapter	Read the files and enter their directories
<code>/opt/appx-chart-examples</code>	The downloaded <code>chart-smoke.php</code> test script used in Chapter 4 and any other example input files you choose to save	Read the files and enter the directory; administrators install or replace these files
<code>/var/lib/appx-charts</code>	Generated PNG images, including the smoke-test output and charts rendered through APPX	Create, read, update and remove chart files, subject to their individual permissions

These locations are examples. Your paths, APPX runtime account and group will probably differ. Substitute your choices throughout the commands and the APPX parameter form. The commands below use account `appx` and group `appxgrp`. For a fresh system, complete the APPX installer in Chapter 5 first so that account and group exist. Check the account that actually runs your APPX sessions; the account used to install the software may be different.

Create the output and example directories

Run these commands from an administrator account permitted to use `sudo`:

```
sudo install -d -o appx -g appxgrp -m 0750 /var/lib/appx-charts
sudo install -d -o root -g root -m 0755 /opt/appx-chart-examples
```

The first command creates the chart-output directory with `appx` as its owner. The second creates the example-script directory with `root` as its owner. Both commands set directory attributes; they do not download scripts, install ChartDirector or create a chart. If a named directory already exists, `install -d` applies the requested ownership and permissions to that directory. Check your choices before running it against an existing installation.

The syntax is the same in both commands:

Part	Meaning in these commands
<code>sudo</code>	Run this command with administrator privileges, needed to create directories under <code>/opt</code> and <code>/var/lib</code> and assign their owners
<code>install -d</code>	Create a directory and any missing parent directories; <code>-d</code> selects directory creation rather than file copying
<code>-o appx</code> or <code>-o root</code>	Set the named directory's owner to the account following <code>-o</code>
<code>-g appxgrp</code> or <code>-g root</code>	Set the named directory's owning group to the group following <code>-g</code>
<code>-m 0750</code> or <code>-m 0755</code>	Set the named directory's permissions to the numeric mode following <code>-m</code>
Final pathname	The directory to create or update

In these numeric modes, the leading `0` marks octal notation. The remaining three digits specify permissions for the **owner**, **owning group** and **everyone else**, in that order. Each digit adds read (`4`), write (`2`) and search/enter (`1`) permissions. For a directory, read permits listing names, write permits adding or removing directory entries, and search permits reaching files through that directory. Creating a file requires both write and search access to its directory.

Mode	Owner	Owning group	Everyone else
<code>0750</code>	<code>7</code> : read, write and search	<code>5</code> : read and search	<code>0</code> : no access
<code>0755</code>	<code>7</code> : read, write and search	<code>5</code> : read and search	<code>5</code> : read and search

With these choices, `appx` can create chart files in `/var/lib/appx-charts`. Members of `appxgrp` can list and enter that directory, but the mode does not give them permission to create files there. The example directory allows APPX to reach scripts installed by an administrator while reserving directory changes to `root`. Files inside either directory have their own permissions: a PHP script invoked through `php` needs read access, and does not need its executable bit set. These directory commands do not recursively change existing files.

Check write access as the APPX account

The command `sudo -u appx test -w /var/lib/appx-charts` runs `test` as `appx`. Here `-u appx` belongs to `sudo` and selects the account; `-w` belongs to `test` and asks whether the path is writable. `test` prints nothing. Its exit status is `0` when the test is true and `1` when it is false; a failure to run `sudo` can also produce a nonzero status.

Use this form to display a result:

```
if sudo -u appx test -w /var/lib/appx-charts; then
    printf '%s\n' 'PASS: appx has write access to the chart directory.'
else
    printf '%s\n' 'FAIL: check the APPX account, directory ownership and permissions.' >&2
fi
```

`if` runs the check. A zero exit status selects the commands after `then`; a nonzero status selects those after `else`. `fi` ends the conditional. `printf` prints the message, and `>&2` sends the failure message to the error output. Resolve a failure before continuing.

This checks reported write access without creating a file. The PHP smoke test in Chapter 4 must still create a fresh PNG as the APPX account, and the APPX session must render its own chart. A successful administrator-shell check does not establish access from a service running in a different SELinux domain; follow the SELinux topic below.

Make PHP available to the APPX process

APPX invokes the command named `php`. Its `PATH` environment variable is the colon-separated list of directories searched for that command, from left to right. For an interactive test:

```
export PATH=/usr/bin:$PATH
command -v php
php --ini
```

`export PATH=/usr/bin:$PATH` puts `/usr/bin` first and retains the existing list through `$PATH`. `export` makes that value available to child processes started from this shell. `command -v php` shows which executable the shell selects; expect `/usr/bin/php` for this recipe. `php --ini` shows its configuration file and additional ini-file locations. Substitute the directory containing your selected PHP executable if it differs.

This `export` changes the current shell and its later child processes. It does not update an APPX session or service that is already running. For a systemd service, set `PATH` in that service's environment while preserving the entries it needs. After editing a unit or drop-in, reload systemd's unit definitions and restart the actual APPX service during maintenance. Your unit name and existing service configuration will differ by installation.

Continue with the SELinux checks below before testing from the APPX service.

SELinux Allowances for Chart Rendering

These instructions assume SELinux is enforcing. PHP ChartDirector runs as a child of APPX and writes a local image file. It does not need an HTTP port or a web-server SELinux boolean. The permissions that matter are the APPX process domain, access to PHP and the ChartDirector libraries, and access to the chart directory.

On fresh Rocky 9 and Rocky 10 systems, the documented PHP installation and the shipped APPX simple-bar chart worked with SELinux enforcing. A systemd-launched APPX session ran in `unconfined_service_t` and rendered successfully. No additional ChartDirector allow module or SELinux boolean was required. Your service may use a different domain; check it before copying a policy change from another installation.

Check the paths used by the failing session

If a chart fails, reproduce it once through the APPX session that failed. Then inspect the process contexts, path labels and recent denials:

```
ps -eZ | grep -E 'appx|php'
ls -ldZ /opt/ChartDirector /opt/ChartDirector/lib \
    /var/lib/appx-charts
ls -lZ /usr/bin/php /opt/ChartDirector/lib/phpchartdir830.dll \
    /opt/ChartDirector/lib/libchartdir.so
sudo ausearch -m AVC,USER_AVC -ts recent -i
```

Use your actual library and chart paths. PHP may have finished before `ps` runs; the audit event still records the denied process, its source context (`scontext`), the target context (`tcontext`), the object class and the denied permission. Match the event's time, executable and path to the chart request before changing policy. A permission error with no matching AVC may be an ordinary ownership, directory-traversal or filesystem permission problem.

Restore an incorrect label

Files copied from a home directory or unpacked in a temporary directory can retain an unsuitable label. Compare their current labels with the policy's expected labels:

```
matchpathcon -V /opt/ChartDirector/lib/phpchartdir830.dll
matchpathcon -V /opt/ChartDirector/lib/libchartdir.so
matchpathcon -V /var/lib/appx-charts
sudo restorecon -Rv /opt/ChartDirector /var/lib/appx-charts
```

`restorecon` applies the file-context rules already installed on your server. It does not grant a new permission. Retry the chart after correcting a label. For different installation paths, substitute those paths in every command.

A service with a confined SELinux domain

Your APPX service may use a site-specific confined domain. Check the actual `scontext` in its denial; a successful test from an administrator's shell does not establish that the service has access.

The service needs to execute its selected PHP CLI, read the PHP configuration and include files, load the ChartDirector native libraries, read the fonts and any chart image assets, and create and update files in its dedicated chart directory. Give chart-output write access only to that directory. PHP and ChartDirector program files need read and execute access, not runtime write access.

Use the file types provided by that service's existing policy. If a dedicated chart-output type already exists, add a persistent mapping for the exact chart directory with `semanage fcontext`, then apply it with `restorecon`. The type name depends on your policy; substituting a web-content type does not establish that APPX can use it. Keep a local policy exception limited to the source domain, destination type, object class and permissions identified in the matching AVC. Review those rules before installing them. There is no single additional APPX or ChartDirector allow module required by every installation.

Rocky 9 compatibility container on Rocky 10

The container command earlier in this chapter uses `:Z` on each dedicated bind mount. Podman assigns a private container label so that the APPX process inside that container can use its data and chart directories while the host remains enforcing. Do not apply that private relabel to a directory shared with unrelated services. Your host paths will differ if you chose another container location.

After rendering, confirm the new PNG exists and review the matching audit interval again. Keep SELinux enforcing while correcting a label or adding a narrowly scoped policy rule.

CHAPTER 3

Install on Windows Server

Install and check the Windows command line runtime.

Install the Current PHP CLI

Use the official PHP 8.5 **x64 Non Thread Safe ZIP**. The `Win32` text in its filename is historical; the `x64` suffix identifies the architecture. Use the matching Microsoft Visual C++ x64 runtime required by that PHP build.

1. Open an elevated PowerShell window.
2. Create `C:\AppxTools\PHP85` and a download directory.
3. Download PHP using the release metadata, verify its SHA-256 checksum, then extract it.

```
$ErrorActionPreference = 'Stop'
$ProgressPreference = 'SilentlyContinue'
[Net.ServicePointManager]::SecurityProtocol = `
    [Net.SecurityProtocolType]::Tls12
$download = 'C:\AppxTools\Downloads'
New-Item $download -ItemType Directory -Force | Out-Null
$all = Invoke-RestMethod `
    'https://downloads.php.net/~windows/releases/releases.json'
$release = $all.'8.5'
$package = $release.'nts-vs17-x64'.zip
$archive = Join-Path $download $package.path
curl.exe -fLsS --max-time 240 `
    ('https://downloads.php.net/~windows/releases/' + $package.path) `
    -o $archive
if ($LASTEXITCODE -ne 0) { throw 'PHP download failed' }
if ((Get-FileHash $archive -Algorithm SHA256).Hash `
    -ne $package.sha256) { throw 'PHP checksum mismatch' }
Expand-Archive $archive 'C:\AppxTools\PHP85' -Force
```

The backtick continues a PowerShell command onto the next line. Keep it as the final character on that line, or enter the command on one line.

The baseline package is `php-8.5.11-nts-Win32-vs17-x64.zip`. Its published SHA-256 is `0ea96e0d2b9b737a6036f05cf4e95c49313faa6d0f27bd97edb2742503f0c043`. The metadata procedure deliberately chooses the current PHP 8.5 maintenance package at installation time.

Install the Microsoft Runtime

Install the current supported **x64 and x86** Visual C++ v14 Redistributables from [Microsoft](#). The x64 package supports PHP. The x86 package supplies the runtime needed by APPX 5.5.4's bundled x86 `libcurl.dll`; x64 alone did not allow that engine to start in the clean Server 2025 test.

```
foreach ($arch in 'x64', 'x86') {
    $runtime = Join-Path $download ("vc_redist.$arch.exe")
    curl.exe -fLsS --max-time 240 `
        ("https://aka.ms/vc14/vc_redist.$arch.exe") -o $runtime
    if ($LASTEXITCODE -ne 0) { throw 'Runtime download failed' }
    $signature = Get-AuthenticodeSignature $runtime
    if ($signature.Status -ne 'Valid' -or `
        $signature.SignerCertificate.Subject -notmatch 'Microsoft Corporation') {
        throw 'Microsoft runtime signature validation failed'
    }
    $setup = Start-Process $runtime `
        -ArgumentList '/install /quiet /norestart' -Wait -PassThru
    $setup.ExitCode
}
```

An exit code of 0 indicates success; 3010 requests a restart. A newer installed runtime can return 1638. Resolve a missing-runtime error before adding ChartDirector. Match each runtime architecture to the process that needs it.

Install and Configure ChartDirector

Use the absolute-path method recommended in the [current ChartDirector PHP installation guide](#). Match the extension to PHP's version, architecture and thread-safety setting.

1. Download the Windows x64 **PHP** archive from the vendor.
2. Save its SHA-256 checksum.
3. Extract it under `C:\AppxTools` so the library directory is `C:\AppxTools\ChartDirector\lib`.
4. Keep that entire directory together. Include `chartdir.dll` and all supplied companion files.
5. For a fresh PHP ZIP installation, copy `php.ini-production` to `php.ini` in `C:\AppxTools\PHP85`. If that PHP installation already has a `php.ini`, edit the existing file instead of replacing it.
6. Add the following settings. The extension path must point to your extracted ChartDirector library. `America/New_York` is an example timezone; use the timezone appropriate to your installation.

```
extension="C:\AppxTools\ChartDirector\lib\phpchartdir850nts.dll"
display_errors=stderr
log_errors=0n
date.timezone=America/New_York
```

If you choose the vendor's alternative bare-filename method, copy the **complete** `ChartDirector\lib` directory contents into PHP's configured `extension_dir`. Keep `chartdir.dll` beside the extension; copying only the extension DLL is insufficient.

Run the checks through the explicit executable:

```
& 'C:\AppxTools\PHP85\php.exe' -v  
& 'C:\AppxTools\PHP85\php.exe' --ini  
& 'C:\AppxTools\PHP85\php.exe' -m
```

The loaded php.ini must be the file just edited. Check the **Loaded Configuration File** line from `--ini`; editing a different file has no effect on this interpreter. A new CLI invocation reads the settings immediately. The vendor's instruction to restart PHP applies to a PHP process that stays running, such as a web PHP handler.

Set Permissions and Service PATH

Create your chart output directory; this example uses `C:\AppxData\Charts`. In its **Properties > Security** tab, give the account running APPX **Modify** access. Give that account **Read & execute** access to PHP and the complete ChartDirector library tree. Your service account may differ from the account used to install the software. Check the APPX service's **Log On** tab before setting permissions. Program directories do not need runtime write access.

Test the session PATH before making a persistent change:

```
$env:Path = 'C:\AppxTools\PHP85;' + $env:Path  
where.exe php  
php --ini
```

Add `C:\AppxTools\PHP85` to the machine PATH through the Windows environment-variable editor or an administrator-managed deployment. Preserve existing entries and check for another `php.exe` earlier in PATH. Restart the affected APPX service and establish a new test login. If the service still inherits the old machine environment, restart Windows during maintenance. A successful PowerShell test does not prove that an already-running service has adopted the new PATH.

Use the APPX 6.2.1 **32-bit or 64-bit** server installer. Keep each APPX installation in its own directory. PHP runs as a separate x64 process; its architecture does not need to match APPX.

CHAPTER 4

Verify PHP Before APPX

Render a fresh image under the APPX runtime identity.

Render the Smoke Test

Start with the vendor's **check installation** script, `ChartDirector/phpdemo/cdinfo.php`. The vendor presents this through its demonstration website; for APPX's CLI use, run the same script directly and save its HTML output privately. Run it from `phpdemo` so its relative include resolves correctly.

On Linux:

```
cd /opt/ChartDirector/phpdemo
/usr/bin/php cdinfo.php > "$HOME/chartdirector-check.html"
```

On Windows:

```
Push-Location 'C:\AppxTools\ChartDirector\phpdemo'
try {
    & 'C:\AppxTools\PHP85\php.exe' .\cdinfo.php |
        Set-Content -Encoding utf8 (Join-Path $env:TEMP 'chartdirector-check.html')
    if ($LASTEXITCODE -ne 0) { throw 'ChartDirector installation check failed' }
} finally { Pop-Location }
```

Open the saved HTML locally. Confirm the ChartDirector version and inspect the diagnostic information. The vendor notes that Boot Log and Font Loading Test entries can be informational. Investigate PHP startup errors or a missing version before continuing. This procedure requires no web server; the following smoke test adds a fresh image and filesystem check.

Download `chart-smoke.php` into the example directory. The [Windows installation helper](#) automates the PHP and ChartDirector setup in Chapter 3 and records its package checksums. Run it in an elevated PowerShell window with an unused installation root. This command line example renders three sales values, 120, 90 and 150, to a new 640 by 400 PNG. It checks the file type and dimensions and prints the PHP version, architecture, configuration path, ChartDirector version and image checksum.

On Linux:

```
php /opt/appx-chart-examples/chart-smoke.php \
/opt/ChartDirector/lib/phpchartdir.php \
/var/lib/appx-charts/smoke-01.png
```

Read this command from left to right:

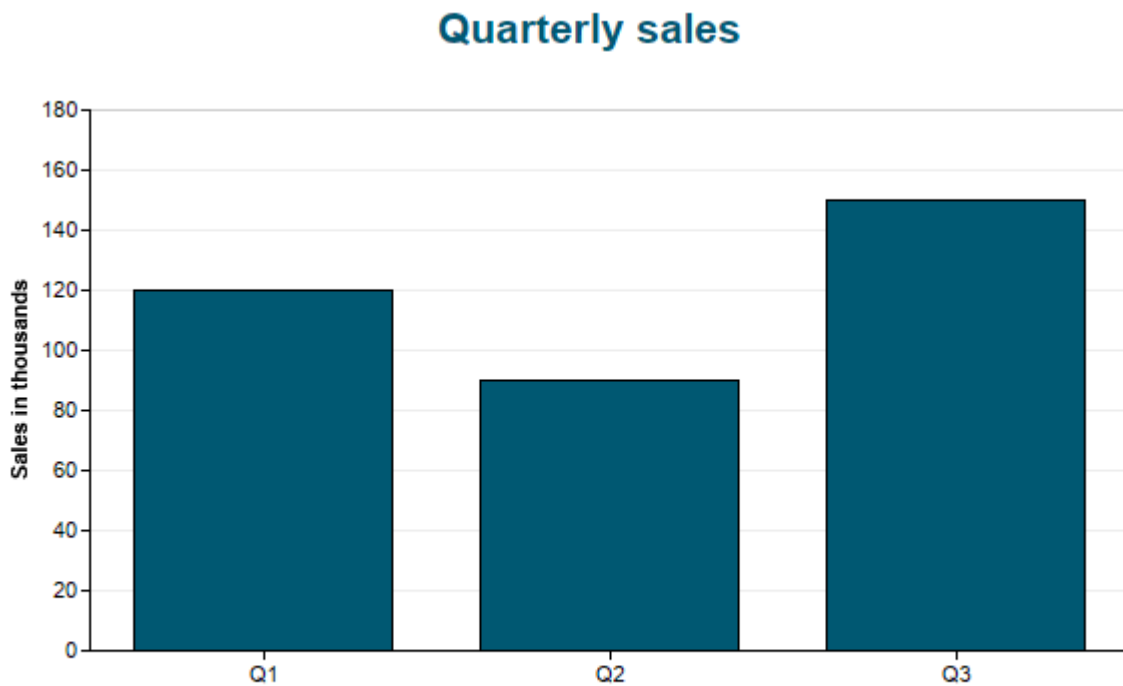
Part	Purpose
<code>php</code>	Run the PHP CLI selected by PATH

Part	Purpose
/opt/appx-chart-examples/chart-smoke.php	Read and execute the downloaded test script; save that file here before running the command
/opt/ChartDirector/lib/phpchartdir.php	First argument to the test script: the vendor's PHP interface file to load
/var/lib/appx-charts/smoke-01.png	Second argument to the test script: the complete filename of the new PNG to create

The backslash (\) at the end of a shell line continues the command on the next line. It must be the final character before the newline. These three lines form one command; you can also type it on one line without the backslashes. If your paths contain spaces, put each complete path in double quotes. Substitute your chosen directories and a fresh output filename.

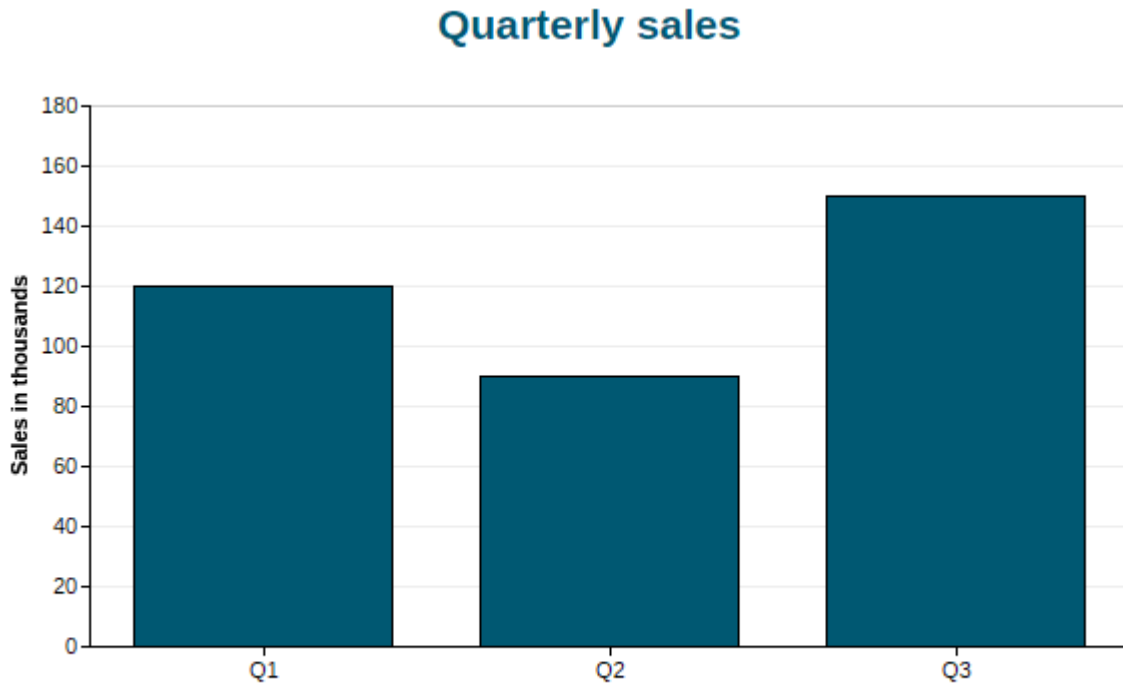
On Windows:

```
& 'C:\AppxTools\PHP85\php.exe' 'C:\AppxTools\ChartExamples\chart-smoke.php' 'C:\AppxTools\ChartDirector\lib\phpchartdir.php' 'C:\AppxData\Charts\smoke-01.png'
if ($LASTEXITCODE -ne 0) { throw 'Chart smoke test failed' }
```



ChartDirector (unregistered) from www.advsofteng.com

Figure 1. PHP CLI smoke chart generated on Windows Server 2025 with PHP 8.5.11 x64 and ChartDirector 7.0



ChartDirector (unregistered) from www.advsofteng.com

Figure 2. The same smoke chart generated on Rocky Linux 10.2 with Rocky AppStream PHP 8.3.33 x64 and ChartDirector 7.0

Choose a fresh output name for each run. The helper refuses to overwrite an existing file so an old image cannot be mistaken for a successful render. Expect `sapi` to be `cli`, `php_bits` to be 64, `thread_safe` to be false, `width` to be 640 and `height` to be 400. Open the image and verify the three bars and their labels.

Test as the APPX Runtime Account

Repeat the same test as the account used by the APPX session, using its selected PHP and configuration. On Linux, an administrator can run:

```
sudo -u appx /usr/bin/php \  
  /opt/appx-chart-examples/chart-smoke.php \  
  /opt/ChartDirector/lib/phpchartdir.php \  
  /var/lib/appx-charts/smoke-appx-01.png
```

On Windows, use an authorized shell or scheduled task running under the actual account. Do not assume a mapped drive, interactive user's PATH or access token is available to the service. Prefer local absolute paths for the first test.

This check isolates PHP loading, fonts and filesystem access. APPX's native chart API adds its own parameter configuration and generated PHP. Complete the next chapter even when the standalone test succeeds.

Purchase and License ChartDirector

ChartDirector can be tested without a license key; the image then includes a yellow evaluation bar. Purchase the license appropriate to your deployment and follow the vendor's [license installation instructions](#).

For a Developer License, place the supplied key in an ASCII text file named exactly `chartdir.lic`. Check that the editor has not appended `.txt`. The vendor's preferred locations are:

Operating system	License file
Linux	<code>/etc/chartdir.lic</code>
Windows	The Windows directory, normally <code>C:\Windows\chartdir.lic</code> ; use <code>\$env:windir</code> to identify it

If you cannot write to that system directory, the vendor permits placing the file beside the **active** `libchartdir.so` on Linux or `chartdir.dll` on Windows. With this cookbook's absolute-path installation, that is the selected `ChartDirector/lib` directory. Give the APPX runtime identity read access and keep the key private. Start a fresh PHP invocation and render a new chart to confirm the evaluation marker has disappeared. Follow the vendor's specific instructions for upgrade keys or redistribution licenses. Preserve the license during upgrades. APPX registration and ChartDirector licensing are separate.

Optional Browser Demonstrations

The vendor's browser demonstrations are optional. A web PHP handler has its own executable, configuration and permissions; verify them separately and still run APPX's CLI test. Keep `phpinfo` diagnostics private and publish only the examples you need. This cookbook requires no HTTP port.

CHAPTER 5

Connect APPX to ChartDirector

Configure the native chart API and run the supplied examples.

Prepare the APPX Installation

Use the official APPX 6.2.1 engine, its matching utility applications and the APPX desktop client. For an existing APPX installation, confirm that applications `0CD` and `1CD` are present, then continue with the desktop walkthrough below. `0CD` contains the chart API and shared parameters; `1CD` contains the examples.

On Windows, run the APPX 6.2.1 **32-bit or 64-bit** server installer as an administrator and choose a dedicated installation directory. On Linux, use the console installer with `-c`. Check the installed engine with `file /path/to/appx` and inspect its dependencies with `ldd /path/to/appx`.

Use the 64-bit APPX engine for native Rocky 10. On Rocky 9, either engine can run natively with its matching libraries. For a 32-bit engine, download `appxsetup-linux-32.sh` instead. A 32-bit engine on Rocky 10 must run inside its prepared Rocky 9 compatibility environment.

The example downloads the installer labelled 64-bit. Verify it against the official release checksum manifest before running it.

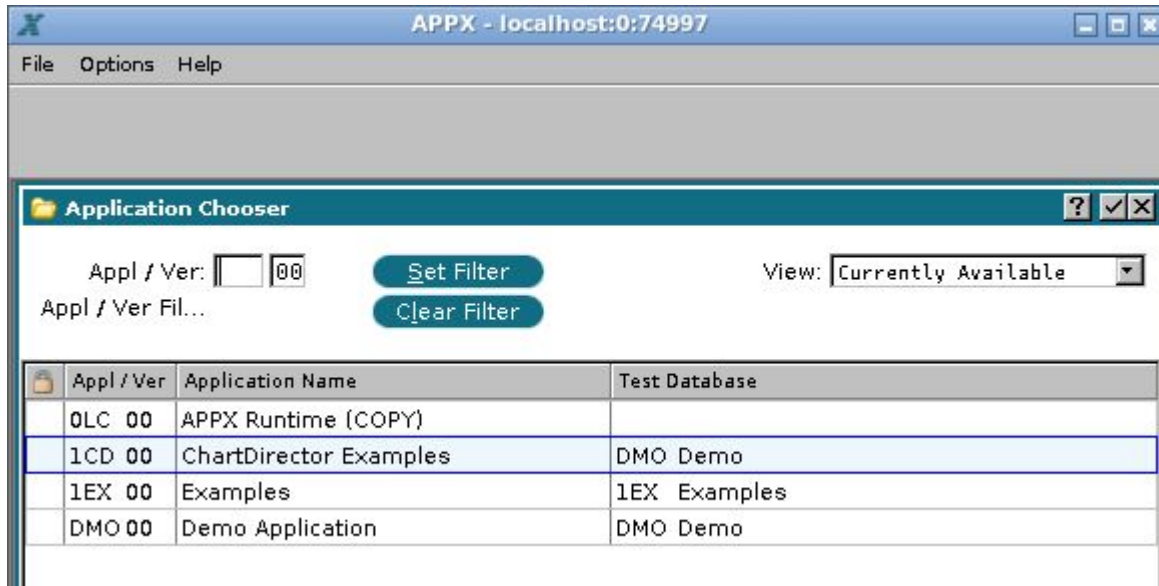
```
cd /tmp
curl -fL -o appxsetup-linux-64.sh \
  https://www.appx.com/ftp/appx/products/appx/server/6.2.1/appxsetup-linux-64.sh
chmod +x appxsetup-linux-64.sh
./appxsetup-linux-64.sh -c
```

Your installation directory will probably differ from the example paths. Complete the installer's prompts, then start APPX for its normal system setup.

For a fresh system, complete normal APPX registration and system setup before continuing.

Open the ChartDirector examples

1. From the APPX opening screen, choose **Design an Application**. In the Application Chooser, select **ChartDirector Examples**, application `1CD`, version `00`, and choose **Open**. The test database in these screenshots is **DMO**. Your database may differ; use the database associated with your installation of `1CD`.



Choose ChartDirector Examples, 1CD 00.

2. On the **Specifications** tab of Application Design, double-click **Menus** under **Form Based Objects**.



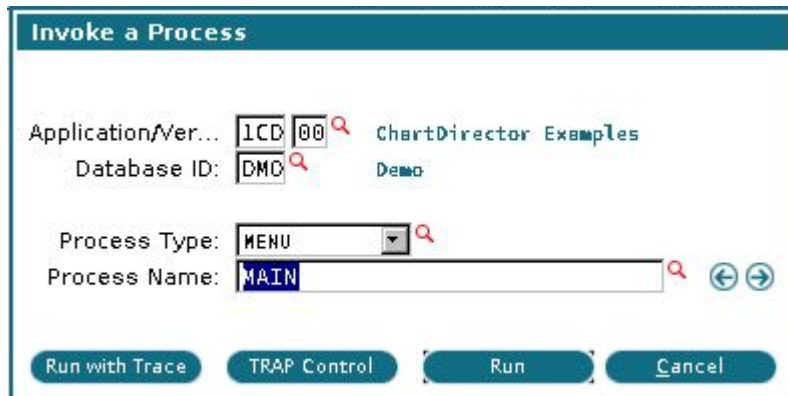
Open Menus in Application Design.

3. Select the MAIN process. Choose **Invoke** on the toolbar.



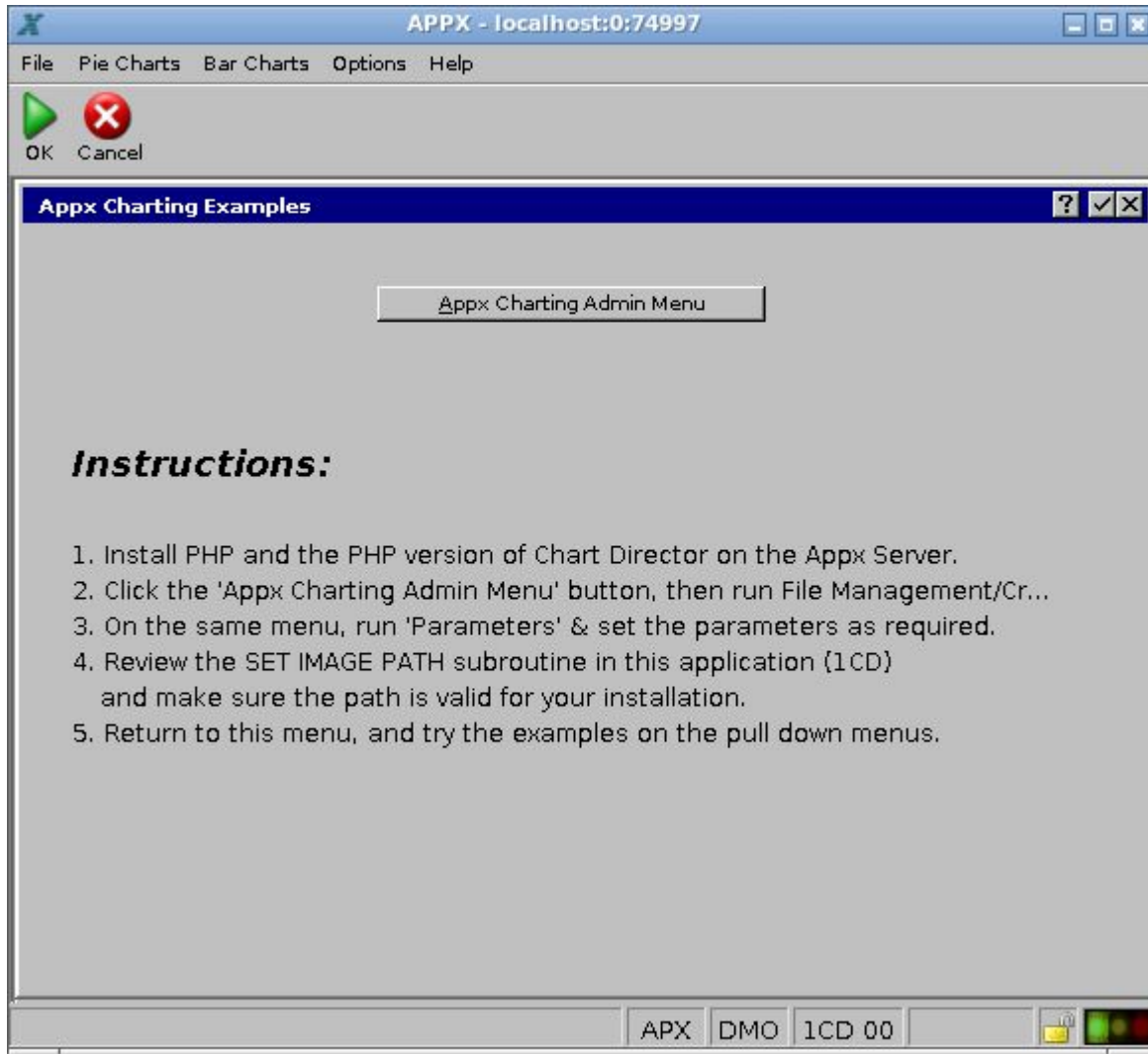
Select MAIN, then choose Invoke.

4. In **Invoke a Process**, confirm application 1CD, version 00, process type MENU, and process name MAIN. Leave the database set to your 1CD test database, then choose **Run**.



Run the 1CD MAIN menu.

The **Appx Charting Examples** menu opens. Keep it open for the next steps.

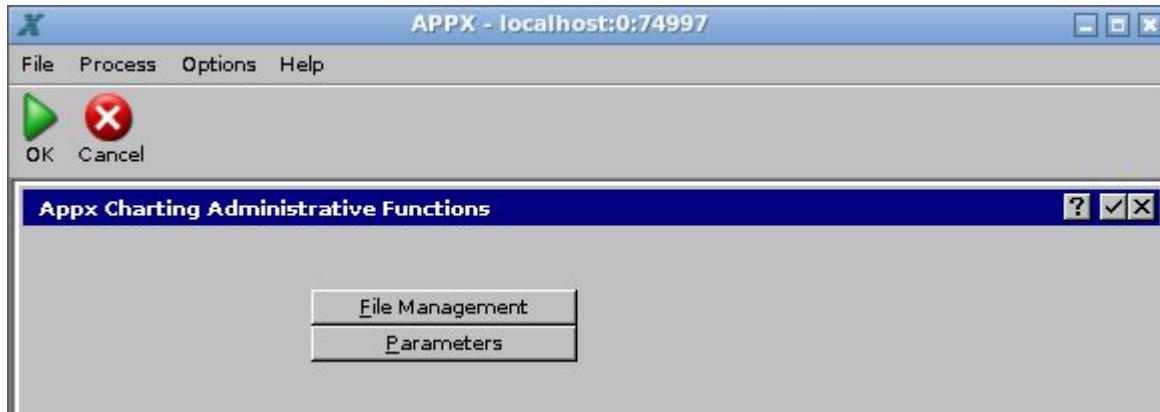


The examples menu opens with its Charting Admin Menu button.

Set the Chart Parameters

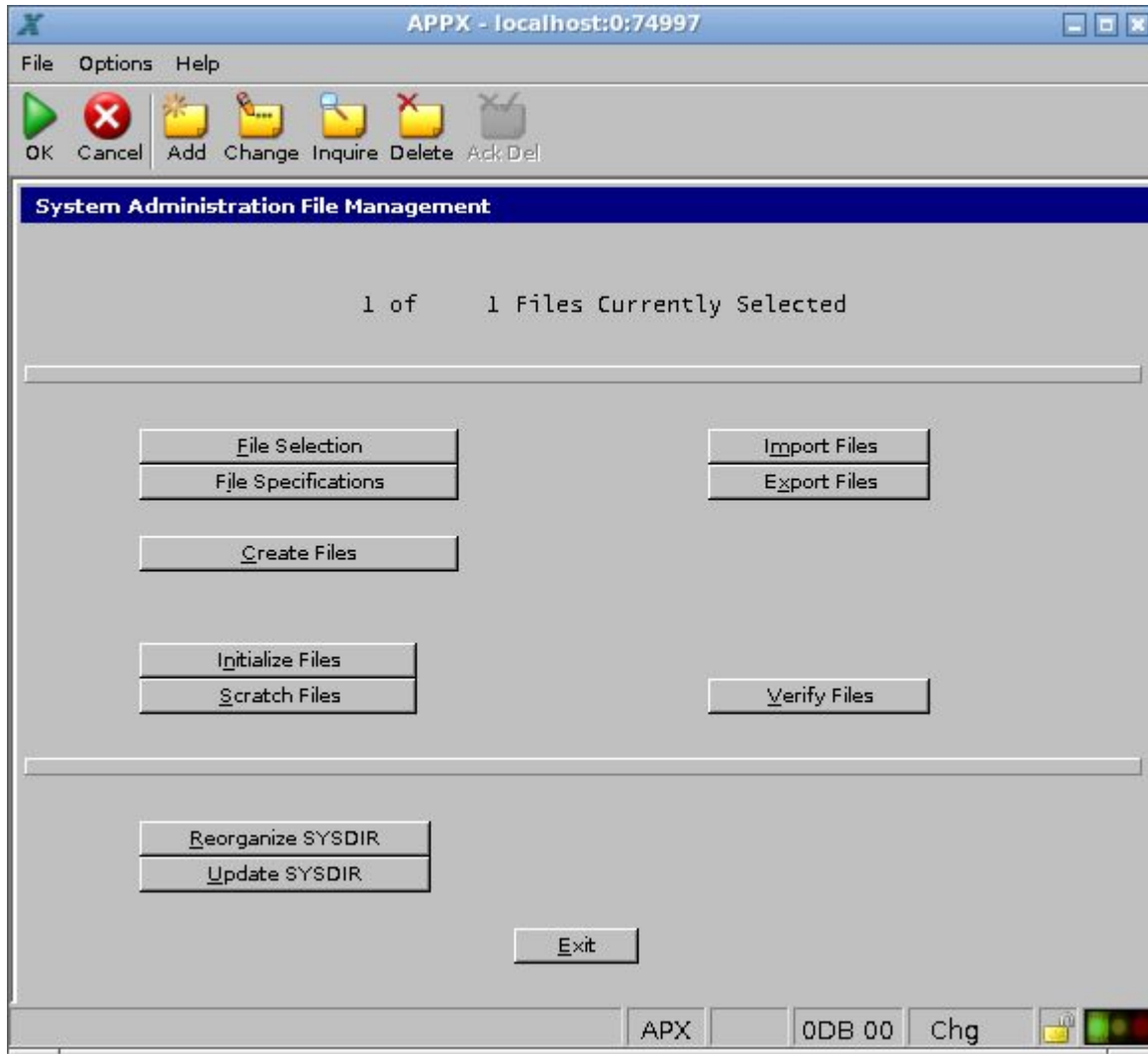
Create the parameter file

1. On Appx Charting Examples, choose Appx Charting Admin Menu. The administration screen contains File Management and Parameters.



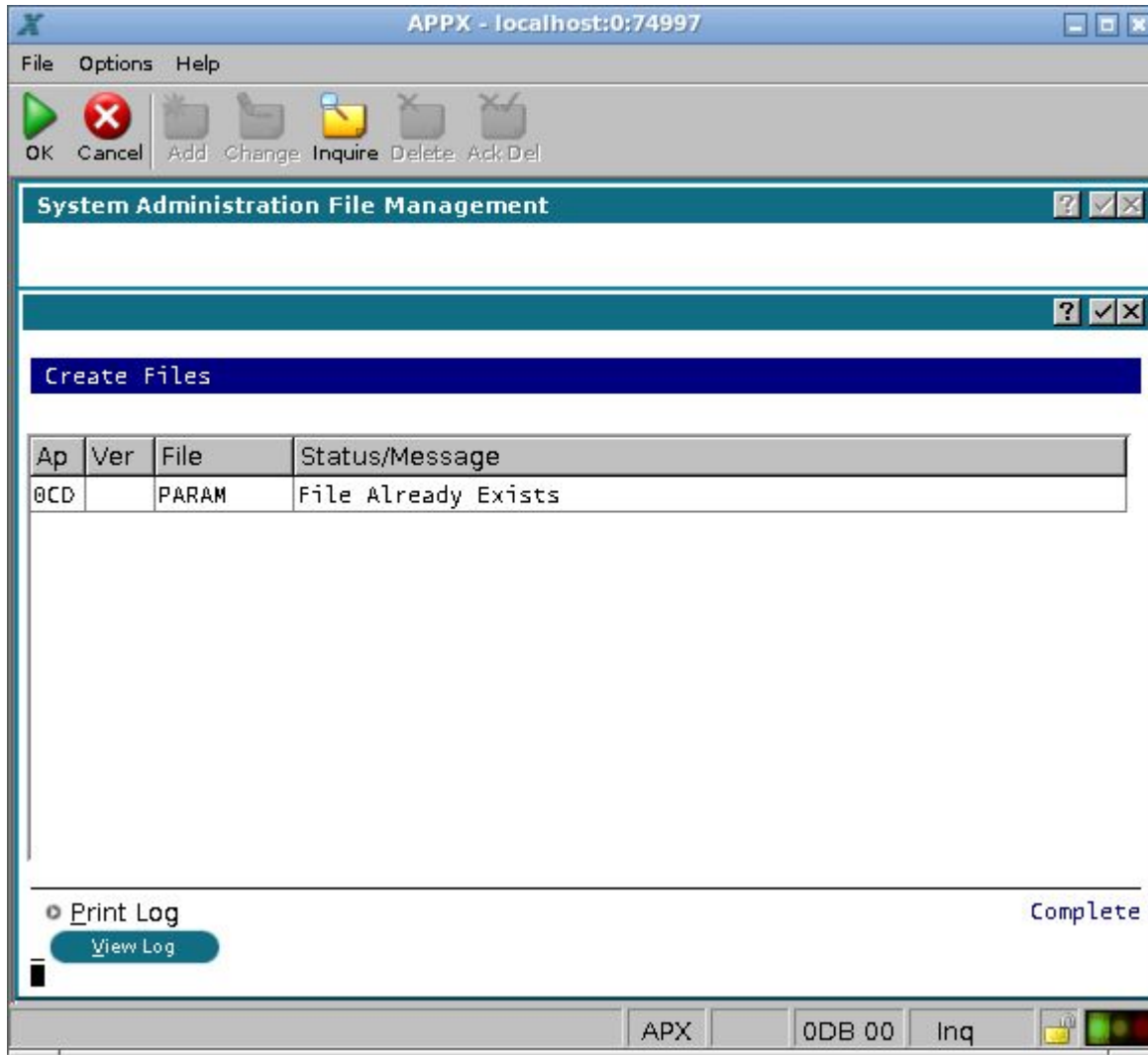
The Charting Admin Menu provides File Management and Parameters.

2. Choose **File Management**. This menu preselects the one **OCD PARAM** file. The screen should show **1 of 1 Files Currently Selected**. If you need to check the file selection, choose **File Selection**, confirm **OCD PARAM**, and return. Choose **Create Files**.



Create the selected OCD PARAM file through File Management.

3. Wait for the result. **Created** confirms a new parameter file. If the file already exists, retain it and its existing settings. File creation also establishes APPX's system-directory entry for the file.



The parameter file already exists in this installation. A fresh installation reports Created.

4. Close the result and leave File Management to return to **Appx Charting Administrative Functions**. Choose **Parameters**. This opens the shared 0CD parameter file used by the chart examples and chart API.

Enter the chart paths

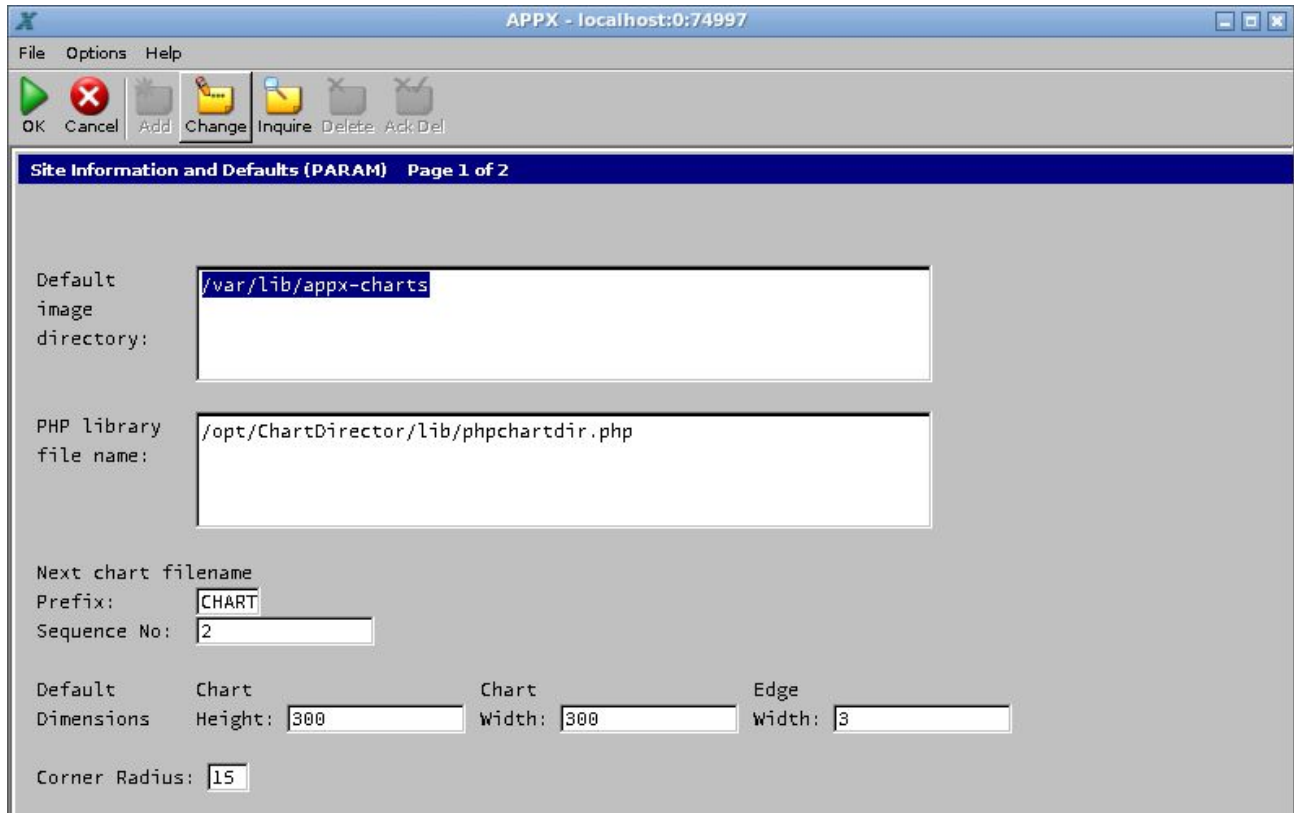
The form has two pages. Enter the paths on the server running APPX. The following table gives the example locations from Chapters 2 and 3; your paths and default chart size may differ.

Parameter	Linux example	Windows example
Default Image directory	/var/lib/appx-charts	C:\AppxData\Charts
PHP Library	/opt/ChartDirector/lib/ phpchartdir.php	C: \AppxTools\ChartDirector\lib\phpchartdi r.php

Parameter	Linux example	Windows example
Finance PHP Library file	<code>/opt/ChartDirector/lib/FinanceChart.php</code>	<code>C:\AppxTools\ChartDirector\lib\FinanceChart.php</code>
Default Chart File Name Prefix	A short prefix such as <code>CHART</code>	A short prefix such as <code>CHART</code>
Default chart width and height	300 and 300 are the shipped defaults	300 and 300 are the shipped defaults

The PHP Library setting is the **full pathname of the PHP include file**, not the extension DLL and not the PHP executable. The Finance library is a separate include used by financial charts; enter it on page 2 of the parameter form. Keep the next chart sequence number consistent with the installation's existing generated files; do not reset a shared live sequence while other sessions are rendering. Save the parameters, then establish a fresh test session.

1. If Parameters opens in **Inquire** mode, choose **Change** on the toolbar. On **page 1**, enter **Default image directory** and **PHP Library file name**. The first is the writable chart-output directory. The second is the complete filename ending in `phpchartdir.php`. The screenshot uses Linux paths; on Windows enter the Windows paths from the table.



APPX - localhost:0:74997

File Options Help

OK Cancel Add Change Inquire Delete Add:Del

Site Information and Defaults (PARAM) Page 1 of 2

Default image directory:

PHP library file name:

Next chart filename

Prefix:

Sequence No:

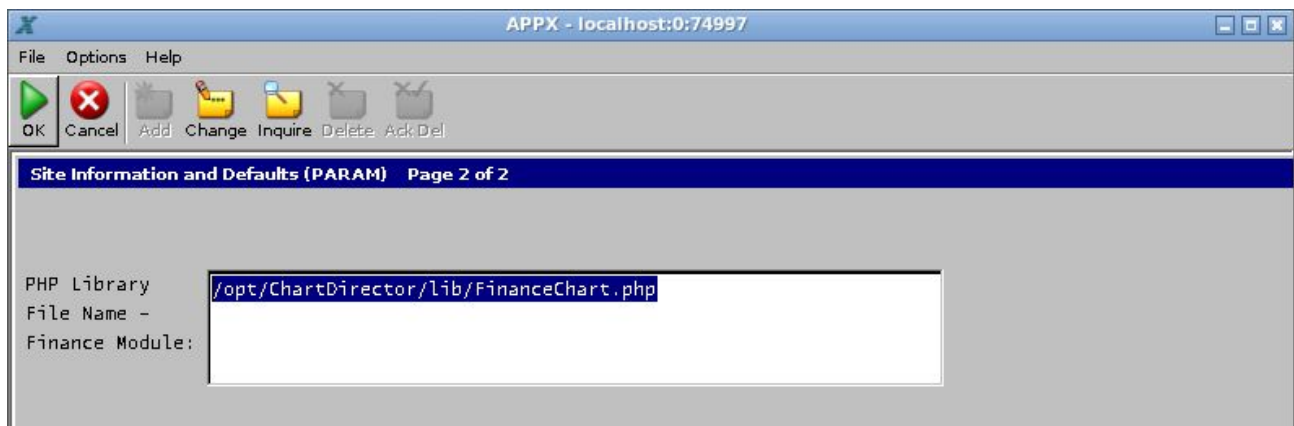
Default Chart Chart Edge

Dimensions Height: Width: Width:

Corner Radius:

Parameter page 1: the chart-output directory and phpchartdir.php filename.

2. Choose **OK** to continue to **page 2**. In the **Finance Module** library field, enter the complete path ending in **FinanceChart.php**. This is a separate PHP include file supplied in the same ChartDirector library directory.



APPX - localhost:0:74997

File Options Help

OK Cancel Add Change Inquire Delete Add:Del

Site Information and Defaults (PARAM) Page 2 of 2

PHP Library File Name - Finance Module:

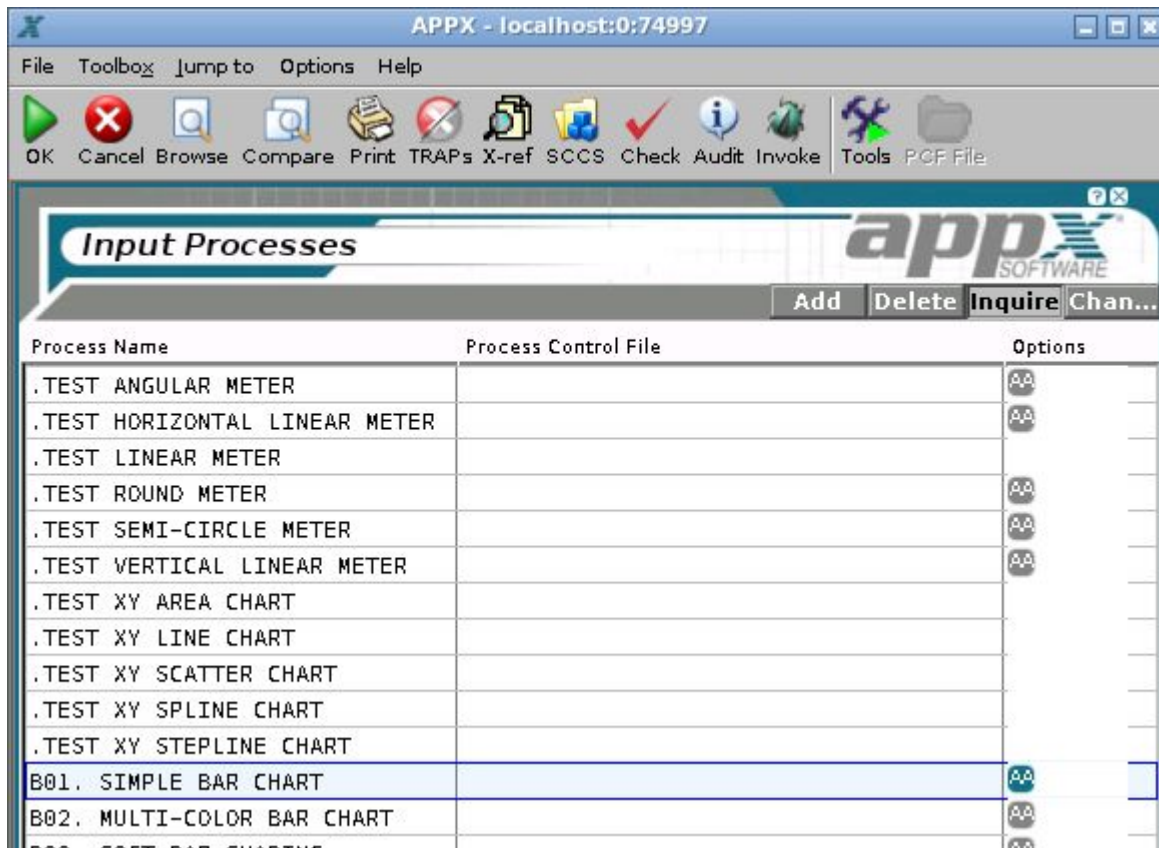
Parameter page 2: the FinanceChart.php filename.

3. Choose **OK** to save page 2 and finish the form. Reopen **Parameters** if you need to check the saved values. Saving only the first page leaves the financial-chart library setting incomplete.

Use a filesystem path to the PHP interface, not a browser URL. Make the chart directory writable by the session account. Confirm both settings in the same execution environment as the APPX engine, particularly when a container is used.

Run a 1CD Example

1. Leave the administration menu to return to **Appx Charting Examples**. Choose **Cancel** on the toolbar to close the examples menu. Close the Menu Processes list to return to **Application Design**.
2. Double-click **Inputs** under **Form Based Objects**. Select **B01. SIMPLE BAR CHART**, then choose **Invoke** on the toolbar.



Select B01. SIMPLE BAR CHART in the 1CD Input Processes list.

3. In **Invoke a Process**, confirm application 1CD, version 00, your test database, process type INPUT, and process name **B01. SIMPLE BAR CHART**. Choose **Run**.

Invoke a Process

Application/Ver... ICD 00 ChartDirector Examples

Database ID: DMO Demo

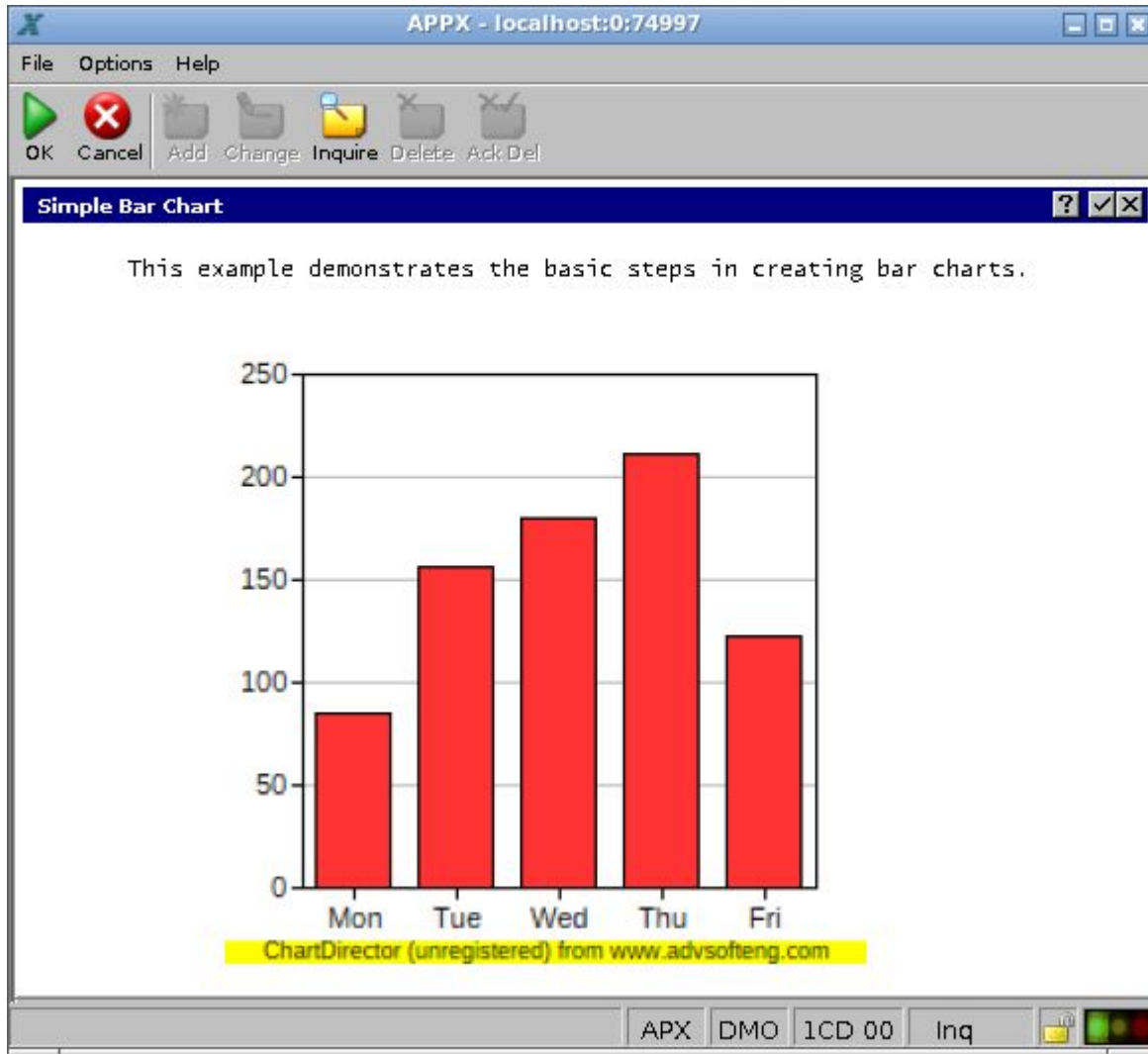
Process Type: INPUT

Process Name: B01: SIMPLE BAR CHART

Run with Trace TRAP Control Run Cancel

Run the shipped simple-bar input process.

4. The sample opens with a bar chart in the APPX screen. Confirm that the bars and labels are visible. This checks the complete path from APPX's chart API through PHP and ChartDirector to the desktop client's server-picture display.



The shipped Simple Bar Chart displayed in the APPX desktop client.

5. Check that a new PNG was written in the chart directory configured in Parameters. If rendering reports an error or the picture is blank, follow **Capture a Render Failure** before changing the sample.
6. Close the sample and run it again. For your own application, also change a data value and check that the new image reflects that change.

The shipped `B01. SIMPLE BAR CHART` sets its own 250 by 250 size. The parameter form's default dimensions do not resize a sample that explicitly sets its dimensions. The standalone PHP test in Chapter 4 creates a separate 640 by 400 image.

If another sample uses icons or background pictures, review `SET IMAGE PATH` in `1CD` and change its image directory to the location of those files on your server. A missing optional icon does not establish that PHP or ChartDirector failed to load.

Capture a Render Failure

Always check `@CD .CHART RENDER` immediately after rendering. A nonblank result means the chart has failed. Inspect `@CD .CHART RENDER ERROR` for details; it can contain PHP diagnostics or HTML. Displaying an old chart after a failed call can conceal the problem.

During development, set `@CD SESSION VIEW PHP` to `Y` and reproduce the failure. APPX downloads and displays the generated PHP. Save it privately, then run it through the same PHP CLI and `php.ini` used by the APPX session. Set the variable back to `N` afterward or end the session; the setting applies to later charts in that session.

```
php -l generated-chart.php
php generated-chart.php
```

Use `-l` to check syntax, then execute the script to test library loading and rendering. A successful syntax check does not exercise ChartDirector. Preserve the first complete diagnostic and its line number when reporting a failure.

CHAPTER 6

Create and Display a Chart

Build a chart and use it in a screen and report.

Build a Small APPX Chart

Start from a shipped 1CD chart close to the required result. The following sequence describes a small bar chart using the documented 0CD API. Define application work fields long enough for the chart name, returned path and result handling; use the parameter field types shown by the matching routine in your installed release.

1. Initialize a one-dimensional dataset through `.DATASET INITIALIZE` and give it a name such as `QUARTER SALES`.
2. Add values 120, 90 and 150 through three `.POINT ADD NEW` calls.
3. Initialize an XY Chart with `.CHART INITIALIZE`. Use a distinct chart name for each independent specification.
4. Add a bar layer through `.XY ADD BAR LAYER`. For this first single-dataset chart, use its documented automatic linking to the current dataset.
5. Set the chart size and plot area, add labels `Q1`, `Q2` and `Q3`, and apply a title.
6. Render to a fresh filename by passing an alpha work field to `.CHART RENDER` with Share `Y`.
7. Check the routine's result and display the returned pathname only after success.

For a process that repeatedly replaces its datasets or handles several series, turn off automatic linking and explicitly manage the data group, dataset template and dataset link. Otherwise a later layer can retain the previously current dataset. The [dataset routines](#) and [bar layer documentation](#) describe the contracts.

Render and Check the Returned Path

Pass the output alpha field with Share `Y` so the routine can return the full PNG pathname. A blank starting value requests a name in the configured chart directory; a full pathname selects an explicit destination. For the first test, use a new filename and a destination you have already tested as writable.

PASS	<chart_pathname>	FIELD	SHARE? Y
SUBR	0CD .CHART RENDER	SUBPROCESS	END? N FAIL 0
IF	0CD .CHART RENDER	NE	

This is the documented call pattern. Replace `<chart_pathname>` with your alpha work field. Put your application's error branch under the nonblank-result test and include `.CHART RENDER ERROR` in its diagnostic. Keep the call in a separate subprocess as shown by the vendor.

Display the Image in an Input

1. Add an alpha work field to the input image and make it long enough for the complete server pathname.
2. Set its GUI control to `PICTURE`.
3. Set Location Type to `SERVER`.
4. Set Position to `EXPAND PROPORTIONAL` and enable the Transparent flag as appropriate for the chart.
5. Size the field to the intended display area.
6. Render before displaying the image and place the successful returned path in this field.

Use proportional expansion to preserve the chart's shape. Test a fresh render on redisplay, leaving the screen and returning, and opening the same screen in a second session. A client-local path and a browser URL are not substitutes for a server image path.

Include the Chart in a PDF Report

Use the same server-picture technique in an APPX OUTPUT image. Render the chart before the image that prints it, pass the successful pathname to the picture field, and produce a PDF through the normal APPX report workflow. Open the saved PDF independently and inspect the chart title, labels, aspect ratio and printed size.

Increase the PNG's dimensions when the printed area needs more detail. For example, a 1,200-pixel-wide chart printed 6 inches wide supplies 200 pixels per inch. Judge small text in the actual PDF at its intended size. Generating a large PNG does not by itself establish that the report layout is readable.

Keep the PNG until report generation and any background printing have finished. Deleting it immediately after the render call can leave the report without its image.

CHAPTER 7

Operate and Maintain the Installation

Manage files, updates and rollback.

Prevent Stale Charts and Filename Collisions

Use a separate output name for each active chart and session. Include a session-specific directory or another controlled unique identifier when your application assigns filenames. Clear the display field or retain an explicit error state after a failed render so a previous successful PNG cannot appear to represent new data.

For repeated renders, verify that the new image contains the new values, not merely that a file exists. Record creation time and a checksum during testing. Two concurrent sessions must not overwrite one another's files or remove files still used by the other session.

Clean Up Generated Files

The APPX chart API leaves generated image files on disk. Remove them only after the screen, report or background task no longer needs them. End of Process can suit a simple input with one chart; multi-chart and deferred-report workflows need their own ownership and retention rules.

Use a dedicated chart directory so a scheduled cleanup can be limited to files created by this application. Exclude files being generated or consumed, and retain failed-script evidence only as long as required for diagnosis. Monitor free space and test cleanup alongside concurrent rendering.

Upgrade PHP or ChartDirector

1. Record PHP, ChartDirector and APPX versions, package hashes, php.ini and chart parameters.
2. Stage the new PHP or complete ChartDirector library in a separate directory.
3. Select the extension for the new PHP minor version and thread-safety setting.
4. Run the standalone test as the APPX identity.
5. Re-run a 1CD sample, an application chart and a PDF report. Compare data, labels and dimensions.
6. Switch the controlled executable or library setting and establish new sessions.
7. Keep the prior installation until rollback has been checked.

A maintenance update within the same PHP minor version uses that extension family: PHP 8.3 uses the 830 extension, and PHP 8.5 uses 850. A move to a different PHP minor release requires a matching vendor extension and repeat the checks. Keep the wrapper `phpchartdir.php` and native libraries from the same ChartDirector distribution.

CHAPTER 8

Troubleshoot by Symptom

Find the failing layer from its diagnostic.

PHP Cannot Be Found

If the error says `php: command not found` or `'php' is not recognized`, inspect the environment of the APPX session and its parent service. Confirm the selected executable with `command -v php` on Linux or `where.exe php` on Windows. Restart the affected service and reconnect after changing its environment. Test through an absolute executable path to distinguish lookup failure from a broken interpreter.

The Extension Does Not Load

Symptom	Check and action
Cannot open shared object or module cannot be found	Confirm the absolute extension path and all companion libraries; inspect <code>ldd</code> on Linux and the required Microsoft runtime on Windows.
Wrong ELF class or not a valid Win32 application	Match the extension architecture to PHP. A 64-bit operating system alone does not identify the running PHP architecture.
Module API mismatch or undefined symbol	Use the extension for the actual PHP minor version and selected TS or NTS build (830 for PHP 8.3; 850 for PHP 8.5). Do not reuse an older extension.
PHP CLI works, APPX fails	Compare <code>PATH</code> , the loaded <code>php.ini</code> , account, current directory and filesystem permissions from the APPX execution context.
Browser demo works, APPX fails	Compare CLI PHP with web PHP. Configure the CLI's <code>php.ini</code> .

Inspect startup errors before changing the application chart definition. An extension that cannot load is below the APPX API layer.

APPX Reports Error Creating Image

Capture `.CHART RENDER` and `.CHART RENDER ERROR`. Enable `SESSION VIEW PHP` temporarily, inspect the generated script and execute it using the same CLI. Check its include path, quoting, chart values, external image paths and output destination. Fix the first diagnostic before treating later messages as separate failures.

Check the generated script against the PHP version actually running. The current ChartDirector PHP package supplies extensions for both PHP 8.3 and PHP 8.5. An obsolete wrapper, mismatched extension or missing dependency should be corrected before changing PHP versions.

The PNG Is Missing or the Wrong Image Appears

Confirm that rendering returned success and that the output file was created during the current request. Check the full returned server path, directory permissions and any security audit messages. Use a fresh output name to eliminate a stale file. Then verify the PICTURE field uses SERVER location and that cleanup has not removed the file before it is displayed or printed.

Fonts or External Images Are Missing

Keep the Linux `fonts` subdirectory beside the ChartDirector native library. Verify that the APPX identity can read the fonts and each image file. Use a simple bar chart without icons as a control. If that works, check `1CD SET IMAGE PATH` and any application-specific image search paths. Do not assume a path from a Windows sample exists on Linux.

A 32 Bit Engine Does Not Start on Rocky Linux 10

A 32-bit engine can report `No such file or directory` even when its file exists, because its ELF interpreter is missing. Run `file` on the actual engine. The name of the installer is not proof of engine architecture. Rocky 10 does not provide the i686 userspace required by this engine. Use the compatibility route in Chapter 2 or the native 64-bit APPX 6.2.1 route where applicable.

APPENDIX A

Reference and Checks

Installation Checklist

Check	Required result
Package identity	Official APPX release and architecture; PHP and ChartDirector versions and hashes recorded
PHP CLI	Correct executable, php.ini, architecture, thread safety and extension
Standalone PNG	New 640 by 400 PNG; values 120, 90 and 150; readable labels
Runtime account	Same test succeeds with the APPX account and environment
SELinux on Linux	Rendering succeeds with SELinux enforcing; any matching denial has been resolved with a path-specific label or narrowly scoped policy rule
Native API	1CD chart returns success and creates a fresh PNG
APPX input	SERVER picture displays the new image with correct proportions
PDF report	Saved PDF contains the correct chart and readable labels
Refresh	Changed values appear after rerendering
Concurrency	Independent sessions keep independent images
Cleanup and rollback	Files remain available until consumed; prior runtime can be restored

The installation examples were checked with Rocky AppStream PHP 8.3.33 x64 NTS and official Windows PHP 8.5.11 x64 NTS. The ChartDirector PHP package is labelled 7.0; its native library reports 7.1.0. The earlier native 1CD checks covered APPX 5.5.4 x86 and APPX 6.2.1 x86 and x64 on the platforms listed in Chapter 1.

The screenshots show APPX 6.2.1 in the desktop client on Linux. Windows uses the same APPX screens; enter the Windows paths shown in the parameter table. Window decorations and fonts can differ with your desktop client and operating system.

Downloads and Sources

The [PHP smoke test](#), [sample values](#), [Windows installation helper](#) and [example README](#) accompany this manual. The PDF embeds these files in its Attachments panel; save them before running an example. Use the instructions above to place them outside a public web directory. Vendor installers remain official external downloads.

Source	Use
APPX downloads	Release-specific installers and checksum manifests
PHP source releases	Current stable PHP baseline

Source	Use
PHP Windows packages	ZIP packages and release metadata
ChartDirector downloads	Current PHP distribution for each architecture
ChartDirector PHP installation	Extension, companion files and license installation
Rocky Linux 10 release notes	CPU baseline and removal of 32-bit compatibility packages
Rocky SELinux documentation	Process and file contexts, persistent labels and audit diagnostics
APPX chart overview	Native chart and dataset workflow
APPX chart rendering	Shared pathname and render result contract
APPX application design manual	Server PICTURE fields and image lifecycle

Version baseline checked October 16, 2026. Record the exact maintenance versions and hashes used for a later installation.

Use the embedded examples

Save these files through the PDF viewer's Attachments panel. They do not run automatically. Read the example README before use. The source packages and vendor installers remain external downloads.

- `README-examples.txt`
- `chart-smoke.php`
- `install-chartdirector-windows.ps1`
- `quarterly-sales.csv`