# Technical Note: SA Automation Platform Extensions (APX)

**I N  T H I S  C H A P T E R**

This section discusses the following topics:

## Introduction to Automation Platform Extensions (APXs)

This section introduces the basic concepts underlying the Automation Platform Extensions (APX).

### Automation Platform Extension (APX) Basics

The SA Automation Platform Extension (APX) feature provides a framework that allows SA professional services teams, third-party integrators, and customers familiar with script-based programming tools such as shell scripts, Python, Perl, and PHP, to extend SA functionality and create applications that are tightly integrated into SA. APXs allow customizations from simple hooks for predefined wizards to stand-alone web-based applications.

SA currently supports two types of APXs:

- Program (OGFS Server script) APX
- Web Application APX

*Program APXs* run in the Global File System (OGFS) and can use all of the OGFS functionality. These APXs provide access to the SA UAPI and to an SA Core's Managed Servers. You can use typical programming practices to leverage the SA UPI and access to a core's Managed Servers to implement functionality. For example, gathering BIOS information from managed servers and populating custom fields using shell commands.

*Web application APXs* allow you to create a web-based application, one in which either an Apache 2.x process or CGI/PHP script is called using GET or POST URL. Web APXs can contain static web resources such as images, and can employ CGI or PHP for dynamic content generation.

Each type of APX will have a runtime environment, a method of passing arguments to it (URL, Positional Argument, and so on), and a defined output method (STDOUT, Typed Results Object, and so on).

APXs allow you to access SA Model Repository data and share that data with web applications or process it with other scripts.

APXs are robust, stable, and secure, for example APXs are:

- Able to survive an upgrade of the SA platform. APXs do not have to be rewritten after an upgrade.
- Managed entities inside the SA Platform. They are registered in the Core's Model Repository.

- Self-describing containers for executable code and meta-data.
- Uniquely identified through versioning.
- Loaded into the Platform as content.
- Able to securely and temporarily escalate a user's permissions beyond the normal defaults during the APX session.
- Scalable within and across cores.
- Schedulable and auditable.

## Program APXs

Program (OGFS Server script) APXs are similar to shell commands. You can invoke them from the OGFS command line and pass input arguments to them using STDIN or command-line arguments. Their output goes to STDOUT and STDERR.

Program APXs are executed inside an OGSH session and have access to all OGSH facilities permissible to the user who invokes the APX. This includes `rosh`, CLI, OGFS, and more. You can write Program APXs using any script-based tool, such as shell script, Python, Perl, and so on.

You can invoke Program APXs from the OGSH command prompt. Typically, Program APXs are executed synchronously, meaning the shell prompt does not return until the Program APX returns. APXs cannot be scheduled as recurring jobs in either the twister or in OGFS.

Program APXs are located in the OGFS directory

`/opsw/apx/bin`

During an interactive OGSH session, a user only sees those Program APXs in `/opsw/apx/bin` that the user has permission to execute. Attempting to invoke a Program APX for which a user has no execution permission results in a `File Not Found` error from the shell.

Program APX can also be invoked by other Web and/or Program APXs. Therefore a CGI program or PHP script from a Web APX can invoke a Program APX.

## Web Application APXs

Web Application APXs are implemented using CGI programs or PHP scripts. These CGI programs and PHP scripts are executed inside a user specific OGSH session. They may access SA facilities such as `rosh`, Unified API (UAPI), CLI, or any commands allowable from within an OGSH session. Web APXs are served by a built-in Apache web server with a PHP module enabled.

There are two ways to access Web APXs: using a stand-alone Web browser such as Internet Explorer or Firefox, or from the SA Client. Microsoft ActiveX is not supported.

Invoking a Web APX from a stand-alone Web browser the first time will trigger a login dialog that requires verification of the SA user credentials. Invoking a Web APX from the SA Client does not require additional login.

Web APXs can be used to build user Interfaces for custom customer applications.

For example, creating custom user interfaces for access to SA functions such as a web page interface that allows a user to do software provisioning on Managed Servers.

To launch APXs using Microsoft Internet Explorer versions 6 and 7 on Windows Server 2003 and 2008 with Enhanced Security Configuration enabled, the OCC website must first be added to Internet Explorer's trusted site list.

## APX User Classes

There are three classes of APX users as shown in Table 1-1:

*Table 1-1: APX User Classes*

| USER | DESCRIPTION |
|---|---|
| **End User** | Is granted permissions to run an APX. This user typically does not have permission to modify an APX or see its content. |
| **APX Developer** | Is granted permissions required to create and publish APXs. This class of users can import and export APXs, and can modify APX content. |

*Table 1-1: APX User Classes*

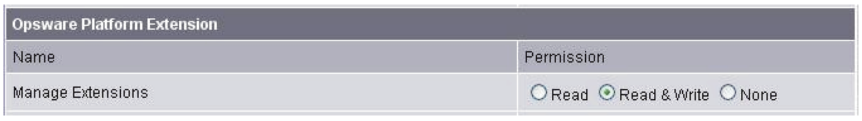
| USER | DESCRIPTION |
|---|---|
| **APX Administrator** | Determines which APXs users are permitted to run. These users assign executable permission to run an APX by managing folder permissions. APX Administrators may not have permission to modify the APX itself, but can have the permission to view APX content in order to determine which APXs to make executable. |

## APX Permissions

APX introduces a new SA Client Feature permission, **Manage Extensions**, in User Group Management. A user group can be given one of the permissions:

- Manage Extensions (Read)
- Manage Extensions (Read & Write)
- Manage Extensions (None)

*Figure 1-1: APX Feature Permissions*



| Opsware Platform Extension | |
|---|---|
| Name | Permission |
| Manage Extensions | Read [✓ Read & Write] None |

These feature permissions apply only to APX developers and administrators, they do not apply to those users who only need to run APXs.

- **Read** permission grants the ability to display the APX source contents or to export (download) the APX source archives.
- **Read & Write** permission grants the ability to modify the contents of an APX in addition to read access.
- **None** permission denies all access to the APX source.

In addition to the SA Client Feature permission, **Manage Extensions**, folder permissions (list, read, write, execute) must be used to calculate which APXs a user has access to.

*Table 1-2: APX Permissions*

| PERMISSION | DESCRIPTION |
|---|---|
| **List** | permission to list the system's APXs. |
| **Read** | permission to view APX content. |
| **Write** | permission to modify APX content and to import/export APXs. |
| **Execute** | permission to run APXs and view APX properties. |

Table 1-3 shows a matrix of how permissions are calculated based on an intersection of the Manage Extensions feature permissions and folder permissions.

*Table 1-3: APX Permission Matrix*

| | READ | READ/WRITE | NONE |
|---|---|---|---|
| **list** | list | list | list |
| **read** | export | export | |
| **write** | export | import, export | |
| **execute** | run | run | run |

Like other SA features, you can grant a user access to an APX and specify to which managed servers and or policies the user can apply the APX.

If a user attempts to access a Web APX which he does not have execution permission, the Web browser will receive an `HTTP 403 Forbidden` return code.

## Permission Escalation

When executing an APX, the user has only the privileges to access resources and operations granted in SA. However, in some cases, it will be necessary to temporarily grant the user *escalated permissions*, privileges beyond the SA privileges, while executing an APX. You can explicitly grant certain privileges to users, over-and-above their default SA privileges, on a temporary basis while running an APX. This permission escalation is transparent to the user running the APX.

For example, you may want the user to be able to run a BIOS information gathering application on a managed server, but the user does not have the permissions granted to do so. You can write an APX that, when run by a user without the privileges required to run the BIOS gathering application, temporarily grants that user the required privileges. The user's privileges return to the default after the APX ends its run.

Privilege escalation is specified in the file `apx.perm` file. For more information, see "The APX Permissions Escalation Configuration File (apx.perm)" on page 22.

## Installing APX Functionality

The APX feature is installed by default with the SA Core installation. No manual configuration is required.

## Extending the APX HTTP Environment

In order to be able to use APX Web programs, you must extend the APX HTTP environment to support PHP or Apache.

You must perform these tasks after all core upgrades

If you have a Multimaster Mesh, these tasks must be performed on each slice in all cores.

### Rebuilding PHP

Perform the following tasks to rebuild PHP to support APX Web applications:

1. Download the PHP source from *http://www.php.net/*.
2. Put the source in a directory on the host where you have your `apxproxy` installed.
3. Enter the following commands:

```
mkdir /build ; cp php-4.4.8.tar.gz /build; cd /build

gzip -dc php-4.4.8.tar.gz | tar xvf -
```

```
cd php-4.4.8

./configure --prefix=/opt/opsware/apxphp

--with-pear=/opt/opsware/apxphp/lib/pear

--with-config-file-path=/opt/opsware/apxphp/lib

--with-apxs2=/opt/opsware/apxhttpd/bin/apxs <any other
options you want>

make clean

make
```

4. Backup Your old copy of `libphp4.so`:

   ```
   cp /opt/opsware/apxhttpd/modules/libphp4.so /opt/opsware/
   apxhttpd/modules/libphp4.so.backup
   ```

5. Copy the new `libphp4.so` file to the `apxhhtps` directory:

   ```
   cp libs/libphp4.so /opt/opsware/apxhttpd/modules/libphp4.so
   ```

6. Ensure that the complete reference library exists in the tool.list:

   ```
   ldd ./libs/libphp4.so
   ```

   For each entry in the output ensure that the file exists in `/etc/opt/opsware/ogfs/tool.list`.

   If an entry does not exist, add it.

7. Backup the `apxphp` folder:

   ```
   mv /opt/opsware/apxphp /opt/opsware/apxphp.orig
   ```

8. Install PHP:

   ```
   make install
   ```

9. Reload and relink the OGFS to make sure anything you added to `/etc/opt/opsware/ogfs/tools.list` shows up in the OGFS:

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/
   tools/reload
   ```

10. Restart `apxproxy`:

    ```
    /etc/opt/opsware/startup/apxproxy restart
    ```

## Rebuilding Apache

Perform the following tasks to rebuild Apache to support APX Web applications:

1. Download the Apache source from *http://www.apache.org/*.
2. Put the source in a directory on the server that hosts the Slice Component bundle.
3. Enter the following commands:

   ```
   mkdir /build; cp httpd-2.2.8.tar.gz /build; cd /build
   ```

   ```
   gzip -dc httpd-2.2.8.tar.gz | tar xf -
   ```

   ```
   cd httpd-2.2.8
   ```

   ```
   ./configure --prefix=/opt/opsware/apxhttpd <any other
   options you want>.
   ```

   HP currently uses:

   ```
   --enable-mods-shared="actions alias auth_basic auth_digest
   authn_file authz_user cgi deflate dir dumpio env expires
   headers ident logio log_config mime negotiation rewrite
   userdir vhost_alias imagemap status"
   ```

   ```
   --disable-dav
   ```

   ```
   --with-port=8021
   ```

   ```
   --with-expat=builtin
   ```

   ```
   --without-pgsql
   ```

   (*On SunOS only*) Enter this command:

   ```
   perl -pi -e 's/#define HAVE_GETADDRINFO 1/#undef HAVE_
   GETADDRINFO/g' ./srclib/apr/include/arch/unix/apr_private.h
   ```

   ```
   make
   ```

4. Make a backup of the `apxhttp` directory:

   ```
   mv /opt/opsware/apxhttpd /opt/opsware/apxhttpd.orig
   ```

5. Install Apache:

   ```
   make install
   ```

6. Reload and relink the new files into the OGFS:

```
/opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/
tools/reload
```

7. The HTTPD and the `.so` files in the modules directory may reference external libraries. These libraries must be visible (or *winked in*) to the OGFS.

   Log in to the OGFS and run LDD on `/opt/opsware/apxhttpd/bin/httpd` and any `.so` file in `/opt/opsware/apxhttpd/modules` and ensure that all the files listed there exist in the OGFS. If they do not, add the files to `/etc/opt/opsware/ogfs/tool.list` (outside the OGFS) and then re-run step 6 until all files are available to `/opt/opsware/apxhttpd/bin/httpd.`

8. You must now rebuild PHP. See “Rebuilding PHP” on page 7.

## APX Structure

An APX has many attributes. Some attributes are maintained by the SA database, such as APX version; while some attributes are self described meta data that is used by the APX runtime. Some of the attributes are not modifiable. For example, after an APX is created, its name and type cannot be changed. However, its display name can be subsequently updated without adversely affecting other APXs which depend on it.

### File Structure

To SA, an APX is a collection of files and directories arrangement of which is application specific. Of course, the arrangement must conform to the general contract of the specific type of APX such that the APX runtime can properly execute it. For example, a Web APX may need a `index.html` or `index.php`. A Program APX may require a shell command provided by the user with the same name as the APX.

### OGFS Integration

The APX infrastructure depends on the OGFS to manage user sessions and it uses the Hub to expose various parts of the APX in the SA file system. The following describe how APX is integrated into the OGFS, and its various applications.

#### *APX Executable Directory*

Program APXs are treated as executable programs in the global shell. These APX are exposed as an executable command in the OGSH. This allows a shell user to invoke the APX as if running a shell command.

The APX executable directory has the following format:

```
/opsw/apx/bin/{apx_name}
```

where `apx_name` is the name of the APX. Running `apx_name` in `/opsw/apx/bin/{apx_name` invokes the current version of `apx_name`.

### *APX Runtime Directory*

The APX Runtime directory is used by the APX runtime to support execution of an APX. The APX Runtime directory must have access to the APX source. In addition, users who have developer privileges, and have read permission to an APX, can also access the APX. The APX Runtime directory is not available for non-APX developers in the global shell.

The APX Runtime directory references the source of the current version of an APX that has been approved. It has the format:

```
/opsw/apx/runtime/{apx_type}/{apx_name}
```

where `apx_type` can be `script` or `web`.

## The APX Tool

The APX Tool is a CLI that is used during an OGFS session to handle all of the management tasks related to APXs. The APX tools has the following functions:

- Creating new APXs
- Importing APXs
- Exporting APXs
- Setting the current version for an APX

### Usage

Invoke the APX Toll from the OGFS command line by using:

```
apxtool [-h | --help] {function} arguments
```

To obtain a complete list of commands and arguments supported by the APX Tool, run the tool from a OGSH command line with no argument.

The APX Tool supports the following major functions:

*Table 1-4: APX Tool Functions*

| FUNCTION | USAGE |
|---|---|
| `delete` | Delete an APX from the SA Library. |
| `export` | Export an APX from the SA Library to a ZIP file. |
| `import` | Import an APX directory or ZIP file into the SA Library. |
| `new` | Create a new APX source directory in the OGFS. |
| `query` | Get information about the APXs in the SA Library. |
| `setcurrent` | Set an APX version as the current version in the SA Library. |

The APX Tool makes use of the following components to perform its functions:

*Table 1-5: APX Component Dependencies*

| COMPONENT | USAGE |
|---|---|
| Global File System (OGFS) | The APX Tool relies on OGFS for access to APX sources using the file system view. |
| Web Services Data Access Engine | The Web Services Data Access Engine provides the APX Management API functions. |
| Pytwist | The Pytwist component provides a means to access the UAPI using python. |

### *APX Tool Options Processing*

Most of the tool's options accept a short format or a long format. The short format is prefixed with a single “ `-` “ character and is immediately followed by an alphanumeric character. For example, “ `-t` “.

A long format is prefixed with two “ `--` “ characters followed by a word. For example, “ `--type` “.

Some options require an argument following the option switch. For example, “ `-t webapp` “. Option arguments can be specified in one of four formats, which are all equivalent. To illustrate, the following commands produce the same result:

```
apxtool query -t webapp
apxtool query -twebapp
apxtool query --type webapp
apxtool query --type=webapp
```

Some options only require typing a minimum number of characters, enough to identify the option argument. For example, in the query function, the `--view` option requires argument “ `list` “, “ `details` “, “ `versions` “. The following commands produce the same result:

```
apxtool query --view=details
apxtool query --view=d
apxtool query -Vdetails
apxtool query -Vd
```

### *Creating a New APX using the APX Tool*

You use the APX Tool to create New APXs.

*Usage*

```
apxtool new [options] {src_dir}
```

where the `src_dir` argument specifies the directory where the source of the new APX is to be created. If this argument is omitted, the current directory is used.

Table 1-6 lists the options that are available when creating a new APX:

*Table 1-6: Create New APX CLI Options*

| OPTION | USAGE |
|---|---|
| `-h, --help` | Show this help message and exit. |
| `-t type, --type=type` | (**Required**) APX type. Valid values are: `script` or `webapp`. For example, -ts for script, -tw for webapp |
| `-u unique name,` `--uniquename=unique name` | (**Required**) APX unique name. |

Table 1-6: Create New APX CLI Options (continued)

| OPTION | USAGE |
|---|---|
| `-n name, --name=name` | (**Optional**) APX display name in a folder. If a name is not specified, but a `uniquename` is specified, the last part of the APX unique name is used as the display name. Note that this name must be unique within the specified folder. |
| `-d desc, --description=desc` | (**Required**) A brief description of an APX. If `desc` is a filename with extension `.txt`, the file is assumed to be a text file and its content is used as the APX description. Otherwise, the `desc` value is used as the description. |
| `-r, --register` | If specified, register the APX into the system. You must also specify `--folder` for this option to work. |
| `-f path, --folder=path` | (**Optional**) SA folder path. `path` can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. This option is only needed if option `--register` is used. |

### *Deleting an Existing APX using the APX Tool*

You can use the APX Tool to delete existing APXs from the SA library.

*Usage*:

```
apxtool delete [options]
```

Table 1-7 lists the options that are available when deleting an APX:

Table 1-7: Delete APX CLI Options

| OPTION | USAGE |
|---|---|
| `-h, --help` | Show this help message and exit. |

Table 1-7: Delete APX CLI Options (continued)

| OPTION | USAGE |
|---|---|
| `-t type, --type=type` | (**Required**) APX type. Valid values are: `script` or `webapp`. For example `-ts` for script. |
| `--id=id` | (**Optional**) APX ID. |
| `-u unique_name,` `--uniquename=unique_name` | (**Optional**) APX unique name. A unique name is a dot separated name that conforms to file system format. It must have at least one dot. Valid characters are: [`a-zA-Z0-9_.`]. **Example**: `com.opsware.security.scan_ports` |
| `-n name, --name=name` | (**Optional**) APX display name in a folder. |
| `-f path, --folder=path` | (**Optional**) SA folder path. `path` can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. |

### *Exporting an APX using the APX Tool*

You can use the APX Tool to export APXs. Export downloads a specific version of an APX source archive file.

*Usage*:

```
apxtool export [options] {target_dir}
```

where the argument `target_dir` is the directory into which the APX source archive file is copied or into which the APX source archive content is expanded, dependent on whether the `--archive` option is specified. If omitted, the current directory is used.

Table 1-8 lists the options that are available when exporting an APX.

Table 1-8: Export APX CLI Options

| OPTION | USAGE |
|---|---|
| `-h, --help` | Show this help message and exit. |

Table 1-8: Export APX CLI Options (continued)

| OPTION | USAGE |
|---|---|
| `-t type, --type=type` | (**Required**) APX type. Valid values are: `script` or `webapp`. For example, `-ts` for script. |
| `--id=id` | (**Optional**) APX ID. |
| `-u unique_name, --uniquename=unique_name` | (**Optional**) APX unique name. A unique name is a dot separated name that conforms to file system format. It must have at least one dot. Valid characters are: [`a-zA-Z0-9_.`].<br><br>**Example**: `com.opsware.security.scan_ports` |
| `-n name, --name=name` | (**Optional**) APX display name in a folder. |
| `-f path, --folder=path` | (**Optional**) SA folder path. `path` can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. |
| `-v version_str, --version=version_str` | (**Optional**) This option specifies which APX version to download. If omitted, the current version is downloaded. |
| `-a, --archive` | If specified, export the APX source in its original source archive as a ZIP or JAR file. |

### *Importing an APX using the APX Tool*

You can use the APX Tool to import APXs. Import publishes a new version of an APX and optionally sets this version as current. If the APX does not exist, this command registers the APX into the system.

*Usage*:

```
apxtool import [options] {apx_src}
```

where `apx_src` can be an archived APX source file with extension `.zip` or `.jar`, or it can be the name of a source APX directory to be published. `apx_src` may be a relative or absolute path. If omitted, the current directory is used.

Table 1-9 lists the options that are available when importing an APX:

*Table 1-9: Import APX CLI Options*

| OPTION | USAGE |
|---|---|
| `-h, --help` | Show this help message and exit. |
| `-c, --setcurrent` | If specified, set the newly published version as the current version of an APX. |
| `--version=version_str` | The new version of this APX. This option must not be used if `version_str` is already specified in `apx.cfg`. If no version is specified, one will be assigned automatically. |
| `-f path, --folder=path` | (**Optional**) SA folder path. `path` can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. |

### *Querying APX Information using the APX Tool*

You can use the APX Tool to query APX information. You can specify additional options to limit resulting APX objects. Multiple occurrences of the same option forms a logical `OR` expression. If no matching result is found, returns `exit code 100.`

*Usage*:

```
apxtool query [options]
```

Table 1-10 lists the options that are available when querying APX information:

*Table 1-10: Query APX Information CLI Options*

| OPTION | USAGE |
|---|---|
| `-h, --help` | Show this help message and exit. |

Table 1-10: Query APX Information CLI Options (continued)

| OPTION | USAGE |
|---|---|
| `-v view, --view=view` | (**Optional**) Select one of the predefined views of the query results. Choices are `list` (default), `details`, and `versions`.<br><br>`list` is a single line representation of APX basic information presented in tabular format.<br><br>`details` is a multiple line representation of APX information.<br><br>`versions` lists all APX versions. You would only need to specify enough characters for the view type; for example, `-vd`, is the same as `-v details`. If the `versions` layout is selected, the query must result in a single APX object. |
| `-t type, --type=type` | (**Required**) APX type. Valid values are: `script` or `webapp`. For example, `-ts` for script. |
| `--id=id` | (**Optional**) APX ID. |
| `-u unique_name,`<br>`--uniquename=unique_name` | (**Optional**) APX unique name. A unique name is a dot separated name that conforms to file system format. It must have at least one dot. Valid characters are: `[a-zA-Z0-9_.]`.<br><br>**Example**:<br>`com.opsware.security.scan_ports` |
| `-n name, --name=name` | (**Optional**) APX display name in a folder. |

*Table 1-10: Query APX Information CLI Options (continued)*

| OPTION | USAGE |
|---|---|
| `-f path, --folder=path` | (**Optional**) SA folder path. `path` can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. |
| `--current` | (**Optional)** if specified, only query APX objects that have a current version set. |
| `--format=format_str` | (**Optional**) This advanced option allows a user to specify custom display format for an APX listing.<br><br>`format_str` is a string containing embedded tag names that are substituted with values at display time. Tag names must have a format of `%(tag_name)`.<br><br>Use a special format string of `__show_tags__` to show a list of supported tag names. |
| `--csv` | (**Optional**) If specified, display result in CSV (comma separated values format). Ignored if `--format` option is specified. |

### ***Setting the APX Version using the APX Tool***

You can use the APX Tool to set an APX version as the current version.

*Usage*:

```
apxtool setcurrent [options] {version_str}
```

where the argument `version_str` is required to uniquely identify an existing version of an APX.

Table 1-11 lists the options that are available when setting an APX version:

*Table 1-11: Setting the APX Version CLI Options*

| OPTION | USAGE |
|---|---|
| `-h, --help` | Show this help message and exit. |
| `-t type, --type=type` | (**Required**) APX type. Valid values are: `script, webapp`. For example, `-ts` for script. |
| `--id=id` | (**Optional**) APX ID. |
| `-u unique_name,` `--uniquename=unique_name` | (**Optional**) APX unique name. A unique name is a dot separated name that conforms to file system format.It must have at least one dot. Valid characters are It must have at least one dot. [`a-zA-Z0-9_.`].<br><br>**Example**: `com.opsware.security.scan_ports` |
| `-n name, --name=name` | (**Optional**) APX display name in a folder. |
| `-f path, --folder=path` | (**Optional**) SA folder path. `path` can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. |

### *Error Handling*

As a command line interface (CLI), the APX Tool conforms to the standard POSIX convention to return 0 on success, and a non-zero value for other errors. The APX Tool produces normal output in STDOUT, and error/warning information in STDERR.

On error, the APX Tool typically returns a descriptive message in STDERR.

Error conditions are typically categorized as shown in Table 1-12:

*Table 1-12: APX Tool Error Conditions*

| RETURN CODE | DESCRIPTION |
|---|---|
| 0 | Success |
| 1 | Syntax or usage error |
| 2 | Permission related error |
| 3 | User canceled operation |
| 4 | Runtime error |

There may be other undocumented exit codes. The only guarantee is that if exit code is 0, the command completes its operation successfully.

## The APX Configuration File (apx.cfg)

All APXs, regardless of type, must have a configuration file, `apx.cfg`. This file contains meta data that fully describes the APX. For simplicity, the `apx.cfg` uses a simple `key=value` property format. Multiple lines are joined together with a line continuation character, " `\` ".

The following describes common attributes for all APX. APX type specific attributes are described in the corresponding APX type functional specifications. Note that some of the attributes may be extracted from the apx.cfg configuration file, and managed in SA. For mutable type of attributes (such as Description), subsequent updates of the apx.cfg will update the SA managed data accordingly.

*Table 1-13: APX Configuration File Attributes*

| ATTRIBUTE | MODIFIABLE? | DESCRIPTION |
|---|---|---|
| **Type** | No | The type of the APX, **WebApp** or **Script**. Once created, you cannot change the APX Type. |

*Table 1-13: APX Configuration File Attributes (continued)*

| ATTRIBUTE | MODIFIABLE? | DESCRIPTION |
|---|---|---|
| **Name** | No | The name of the APX. The Name must be unique. This name will be used as the filename for the APX as it appears in the OGFS. The Name together with the Type forms a key that uniquely identifies an APX. Once created, the Name cannot be changed. Since this name is used in the file system, it must conform to the file system naming specification. Generally, name should be in ASCII. |
| **Display Name** | Yes | This is the label and may contain multi-byte characters. Labels can be changed at anytime. This Display Name will be listed in the SA Client APX Folders. |
| **Description** | Yes | A text description of what the APX does. |
| **Usage** | Yes | A text description describing how to use the APX. |

### The APX Permissions Escalation Configuration File (apx.perm)

You use the file, `apx.perm`, to specify permission escalation rules. If this file does not exist, or if it contains no escalation permissions, the APX will run with the user's default permission context.

When a new APX is created using the APX Tool's **New** command, it generates certain default files, including a default `apx.perm` file which, by default, has no escalation permissions defined. The default file does contain some commented out examples which an APX developer can use as templates.

There are three ways to specify escalations:

#### *No Escalation*

The escalations attribute is not specified. The APX runtime uses the current user privilege to execute an APX. If an APX invokes privileged operation which a user does not have, APX execution will terminate with an error.

### *All Permissions*

This is a special privilege that temporarily grants all operation permissions to a user. It is intended for development or demo use only. This is a useful tool for speedy proof of concept, or demo, without worrying fine grain permission tuning. It is a poor choice for a production environment due to its lack of security.

To grant all permissions, edit file `apx.perm` with a macro that matches all features with wildcard characters. For example:

```
use_feature(name="*")
```

### *With Escalation*

Specify a list of predefined common operations in the `apx.perm` file. When executing the APX, the APX runtime temporarily grants these permissions to the APX. SA has a comprehensive list of feature and resource permissions. To simplify the task of escalating related feature, one can use wildcard characters to match groups of related features. For example:

```
@use_feature(name="Application.*")
```

APX Directories, Configuration Files, and Sample APXs

## Tutorial: Creating a Web Application APX

This tutorial demonstrates how to create, publish, and run a simple web application APX named `myfirstapp`.

Running the default version of the APX created during this tutorial displays the output of the PHP command, `phpinfo`. Later in the tutorial, you are shown how to modify the PHP code so that it displays a list of managed servers. Because the tutorial provides the source code, prior knowledge of PHP is not required.

This tutorial contains the following sections. Be sure to complete the tasks in each section in the order shown below:

9 Set Permissions and Create the Tutorial Folder

10 Create a New Web Application

11 Publish the New Web Application

12 Run the New Web Application

13 Modify the Web Application

14 Run the Modified Web Application

*Tutorial Prerequisites*

In order to complete this tutorial, you must have the following capabilities and environment:

- Access to SA 7.0 or later in a development environment.
- The ability to log on to SA as `admin` or as another member of the **Super Administrators** group. (In version 7.0, the **Administrators** group was renamed to **Super Administrators**.) Logging on as `admin` enables you to set permissions.
- The ability to log on to SA as a user who belongs to the **Advanced Users** group.

  Advanced users have permission to create and run the web application. In the example commands shown in this tutorial, the name of this user is `jdoe`.
- An understanding of how to set client feature permissions in the SAS Web Client.

  For more information about permissions, see the User and Group Setup chapter of the *SA Administration Guide*.
- An understanding of how to create folders in the SA Client

  For details on folders, see the Software Management Setup chapter of the *SA Policy Setter's Guide*.
- An understanding of how to open a Global Shell session.

  For instructions, see the Global Shell chapter of the *SA User's Guide: Server Automation*.
- An understanding of basic Unix commands such as `ls` and `cd`.
- Experience developing web applications that run on HTTP servers.

## Set Permissions and Create the Tutorial Folder

1 Log on to the SAS Web Client as `admin` and verify that the **Advanced Users** group has the following permissions:

  – Manage Extensions: Read & Write

  – Allow View Extension Sources: Yes

  These permissions reside on the Client Features tab in the SAS Web Client.

2. Log on to the SA Client as a member of the **Advanced Users** group and create the following folder:

   ```
   /Development/My App
   ```

   Later in the tutorial, you will upload a web application into the `My App` folder. In the non-tutorial environment, the name of this folder is arbitrary. You can create or choose any other folder to contain your web applications.

3. Exit the SA Client.

4. Log on to the SA Client as `admin` and open the **Folder Properties** of the `My App` folder.

5. On the **Permissions** tab of **Folder Properties**, make sure that the **Advanced** Users group has the following permissions:

- List Contents of Folder
- Read Objects Within Folder
- Write Objects Within Folder
- Execute Objects Within Folder

6. Exit the SA Client.

## Create a New Web Application

1. Open a Global Shell session as an SA user who belongs to the **Advanced Users** group.

2. In your core's OGFS home directory, create a directory named `myfirstapp` and then change to that directory:

   ```
   $ mkdir myfirstapp
   $ cd myfirstapp
   ```

   The web application files will be saved to the `myfirstapp` directory.

3. Using the APX Tool command, `new`, create the directory structure and default files for the web application as shown below.

   ```
   $ pwd
   /home/jdoe/myfirstapp
   $ ls
   $
   $ apxtool new -tw  -d "This is my first app." \
   ```

```
-u com.opsware.jdoe.myfirstapp
Create APX 'myfirstapp' source directory /home/jdoe/
myfirstapp? Y/N y
Info: Successfully created APX 'myfirstapp' source
directory: /home/jdoe/myfirstapp.
```

The `-tw` option indicates that the APX type is a web application, `-d` specifies a description, and `-u` specifies a unique name for the application.

If you forget to `cd` to your new subdirectory, the `apxtool new` command displays the following error:

```
Error: Current directory /home/jdoe is not empty,
and cannot be used an APX source directory.
```

For more information about the `apxtool new` command options, see the online help:

```
$ apxtool new -h
```

4. List the files created by the `apxtool new` command:

```
$ pwd
/home/jdoe/myfirstapp
$ ls -R
.:
cgi-bin  images  index.php  APX-INF

./cgi-bin:

./images:

./APX-INF:
description.txt apx.cfg  apx.perm  usage.txt
```

5. Display the contents of the default `index.php` file:

```
$ cat index.php
<?php

// Show information about PHP
phpinfo();

?>
```

As with other web applications, you can replace the `index.php` file with an `index.html` file. However, this tutorial uses the `index.php` file, which you will modify in a later section.

6 Examine some of the files in the `APX-INF` directory.

The `APX-INF` directory contains information that is specific to APX web applications. As shown by the following `cat` command, the `description.txt` file holds the text you specified with the `-d` option of `apxtool new`.

```
$ ls APX-INF/
description.txt  apx.cfg  apx.perm  usage.txt
$ cat APX-INF/description.txt
This is my first app $
```

The following `grep` command shows some of the properties in `apx.cfg`, the APX configuration file. The values for `type` and `uniquename` result from the `-t` and `-u` options of the `apxtool new` command.

```
$ grep "=" APX-INF/apx.cfg
type=webapp
name=myfirstapp
unique_name=com.opsware.jdoe.myfirstapp
enable_live_connect=yes
```

### Publish the New Web Application

Publishing the web application performs the following actions:

- Installs the web application on an HTTP server within SA.
- Imports the web application into a folder that appears in the Library of the SA Client.
- Assigns a version number to the web application.

Enter the `apxtool publish` command and respond to the prompts with `y`, as shown below. The `-f` option specifies the folder in the SA Client where the web application will reside. The `-c` option sets the current version of the web application.

```
$ pwd
/home/jdoe/myfirstapp
$
$ apxtool publish -f "/Library/Development/My App" -c
Do you want to publish current directory: /home/jdoe/
myfirstapp? Y/N y
APX with unique name 'com.opsware.jdoe.myfirstapp' does not
exist.
Register it into the system? Y/N y
Info: Successfully registered APX 'myfirstapp'
(14753430061).
Info: Successfully published a new version '1' for APX
'myfirstapp'
Info: Successfully set APX 'myfirstapp'(14753430061) current
version as '1'.
```

### Run the New Web Application

Now that you have published the web application, you are ready to run it from the SA Client, just as an end-user would.

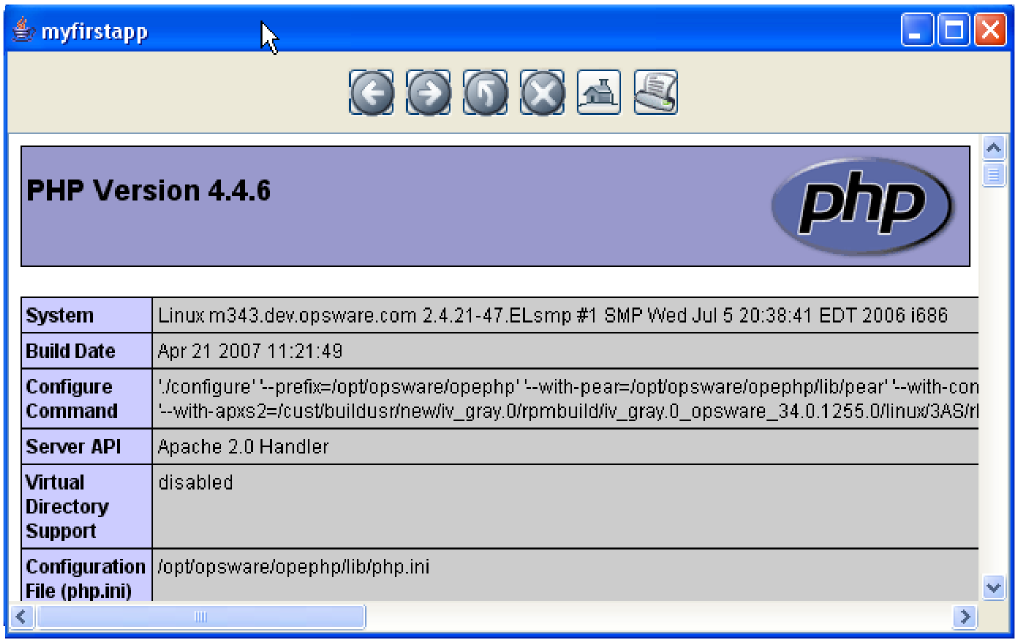
1. Log on to the SA Client as a user who belongs to the **Advanced Users** group.
2. In the Library, navigate to the following web application:

   `/Development/My App/myfirstapp`

   If you do not see `myfirstapp`, make sure that you have the necessary permissions as described in "Set Permissions and Create the Tutorial Folder" on page 24.
3. To run the web application, double-click `myfirstapp`.

The window shown in Figure 1-1 appears. The web application displays the information generated by the `phpinfo` statement of the `index.php` file.

*Figure 1-1: Web Application Version 1*



myfirstapp

**PHP Version 4.4.6**

| | |
|---|---|
| **System** | Linux m343.dev.opsware.com 2.4.21-47.ELsmp #1 SMP Wed Jul 5 20:38:41 EDT 2006 i686 |
| **Build Date** | Apr 21 2007 11:21:49 |
| **Configure Command** | './configure' '--prefix=/opt/opsware/opephp' '--with-pear=/opt/opsware/opephp/lib/pear' '--with-con '--with-apxs2=/cust/buildusr/new/iv_gray.0/rpmbuild/iv_gray.0_opsware_34.0.1255.0/linux/3AS/rl |
| **Server API** | Apache 2.0 Handler |
| **Virtual Directory Support** | disabled |
| **Configuration File (php.ini)** | /opt/opsware/opephp/lib/php.ini |

## Modify the Web Application

Running the default `index.php` file is a good way to check your development environment, but it does not take advantage of SA functionality. In this section, you modify the `index.php` file so that it lists the names of servers managed by SA.

1. In the Global Shell session, locate the `index.php` file of the web application.

```
$ cd /home/jdoe/myfirstapp
$ ls
cgi-bin   images   index.php   APX-INF
```

2. Open the `index.php` file in a text editor such as `vi`.

3. Replace the contents of `index.php` with the following lines:

```
<html>
<head>
<title>Servers</title>
</head>
<body>
<p>List of servers:</p>
```

```
<?php
passthru("ls /opsw/Server/@");
?>

</body>
</html>
```

The preceding `passthru` statement runs the `ls` command, passing `stdout` (without reinflates) back to the web page. The `ls` command lists the names of managed servers as they appear in the OGFS.

4. Save the `index.php` file and exit the text editor.

5. Publish the modified web application.

   The following `apxtool publish` command sets the current version to 2. The `-F` option suppresses the confirmation prompts.

```
$ apxtool publish -f "/home/jdoe/myfirstapp" \
-c --version=2 -F
Info: Successfully published a new version '2' for APX
'myfirstapp'
Info: Successfully set APX 'myfirstapp'(14753430061) current
version as '2'.
```

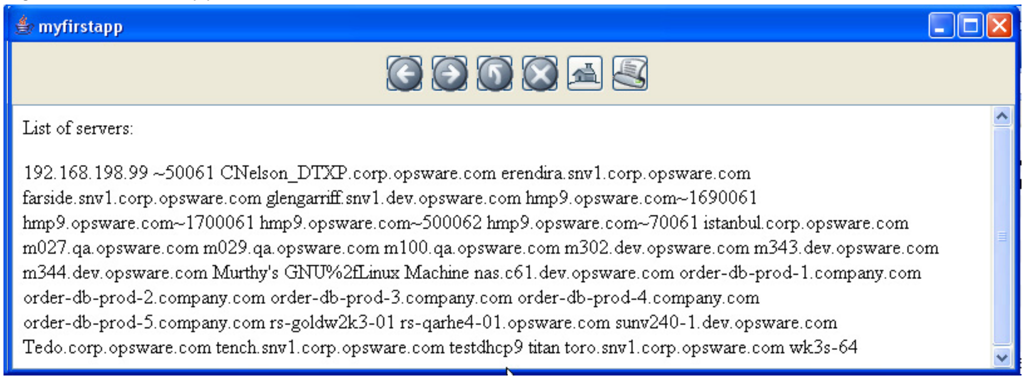
### Run the Modified Web Application

1. In the SA Client, refresh the view of the `My App` folder, which should now contain two versions of `myfirstapp`. Version 2 should be the current version.

2. Double-click version 2 of `myfirstapp` to run it.

Figure 1-2 shows the modified web application, which displays the output of the `passthru` statement of the PHP script. Note that the `passthru` statement removes the line feeds that separate the server names.

*Figure 1-2: Web Application Version 2*



```
List of servers:

192.168.198.99 ~50061 CNelson_DTXP.corp.opsware.com erendira.snv1.corp.opsware.com
farside.snv1.corp.opsware.com glengarriff.snv1.dev.opsware.com hmp9.opsware.com~1690061
hmp9.opsware.com~1700061 hmp9.opsware.com~500062 hmp9.opsware.com~70061 istanbul.corp.opsware.com
m027.qa.opsware.com m029.qa.opsware.com m100.qa.opsware.com m302.dev.opsware.com m343.dev.opsware.com
m344.dev.opsware.com Murthy's GNU%2fLinux Machine nas.c61.dev.opsware.com order-db-prod-1.company.com
order-db-prod-2.company.com order-db-prod-3.company.com order-db-prod-4.company.com
order-db-prod-5.company.com rs-goldw2k3-01 rs-qarhe4-01.opsware.com sunv240-1.dev.opsware.com
Tedo.corp.opsware.com tench.snv1.corp.opsware.com testdhcp9 titan toro.snv1.corp.opsware.com wk3s-64
```