

Apax Cheat Sheet

Scope

The following information are valid for : **Apax version : 3.2.X**
Pls. select another github tag (moving forward) in order to switch description for another Apax feature-version you may use instead.

[01] General

Apax

command	short	description
apax info		Returns a bunch of Apax meta-information, just like local and online-available apax versions, protected scopes etc.
apax --version	-v	Returns the version of the installed Apax.
apax self-update <VERSION>		Update Apax itself to a specific version. If not provided the latest one will be installed.
apax self-update --force-latest	-f	Force an Apax update to its latest version. Be aware of including breaking changes.
apax remove-versions --version=<VERSIONS>		A space separated list of versions, to remove. If not provided all versions other then the current one will be removed.
apax config set <SETTING> <VALUE> --registry <URL>		Set global Apax settings on your system in order to alter the default config, just like "token", "scope", "region".
apax config reset <SETTING> <VALUE> --registry <URL>		Reset the edited global Apax settings on your system in order to fall back to the default.

Login

command	description
apax login	Interactive login.
apax login --registry <URL> --password <ACCESS-TOKEN>	Login to a custom registry e.g. https://npm.pkg.github.com/ .

A login might be required to install dependencies from registries, where an authentication is required.

Licensing

command	description
apax update-license	Update local offline license file for Apax tools.

The license file automatically will be updated when successfully executing `apax install`, `apax update` or `apax add`.

[02] Setup workspaces

command	description
<code>apax create</code>	Create a new project interactively.
<code>apax create <TEMPLATE-NAME> <YOUR- WORKSPACE-NAME></code>	Create a project from a specific template, which is on the default registry. Like : "app" / "lib" / "catalog" etc.
<code>apax create <TEMPLATE-NAME> -- registry <URL> <WORKSPACE-NAME></code>	Create a project from a template, which is not on the default registry.

Example for `apax create` command

```
apax create @simatic-ax/ae-json-library --registry https://npm.pkg.github.com  
myjsonapp
```

Options

create arguments	short	description
<code>--postcreate</code>	<code>-p</code>	Run an existing <code>postcreate</code> script during creation.
<code>--here</code>		Create the new project directly in the current directory, instead of using a new directory.
<code>--no-git</code>		After creation skip the default initialization of a git repository.

Example for `postcreate` script

Command: `apax create my-template --registry=https://npm.pkg.github.com/ --postcreate` This command execute the `postcreate` script, when it is defined in the template `my-template`. In this example, the dependencies will be installed automatically.

```
scripts:  
  postcreate: apax install
```

[03] Manage dependencies

Discover dependencies

command	description
---------	-------------

command	description
<code>apax list --package <NAME> --version <VERSION></code>	List information about a specific version of a package.
<code>apax list --keyword</code>	List all packages containing a specific keyword in their apax.yml.

Install dependencies

command	short	description
<code>apax install</code>		Installs dependencies based on apax.yml manifest and creates a apax-lock.json file. If a lock file exists dependency information will be used for installation except when apax.yml defines a specific version to be used. For details see below
<code>apax install --immutable</code>	<code>-i</code>	Performs a clean & reproduceable install by clearing .apax folder before installing dependencies from scratch. Lockfile will be used as source for dependency metadata. If apax.yml and Lockfile are out of sync (apax.yml defines different version than apax-lock.json) the installation will fail with non-zero exit code.
<code>apax install --copy-local</code>	<code>-c</code>	Copy packages instead of linking to the global cache.
<code>apax install --redownload</code>	<code>-r</code>	Redownload packages, even if they are already available in the global cache.
<code>apax install --catalog</code>		Applies the catalog(s) dependencies-filter to the project. Keeps the same versions of dependencies if they fit within the range the catalog allows.
<code>apax install --catalog --strict</code>		Applies the catalog(s) dependencies-filter to the project. Changes the dependencies also mentioned in the catalog to the exact version defined in the catalog.

Dependencies will be installed according to semantic versioning.

apax.yml	apax-lock.json	available versions on registry	apax install option	installed version
2.0.1	-	2.0.15, 2.5.26, 4.0.0	-	2.0.1
^2.0.1	-	2.0.15, 2.5.26, 4.0.0	-	2.5.26
~2.0.1	-	2.0.15, 2.5.26, 4.0.0	-	2.0.15
^2.0.1	2.0.1	2.0.15, 2.5.26, 4.0.0	-	2.0.1
^2.0.1	2.0.1	2.0.15, 2.5.26, 4.0.0	--immutable	2.0.1

apax.yml	apax-lock.json	available versions on registry	apax install option	installed version
2.0.15	2.0.1	2.0.15, 2.5.26, 4.0.0	-	2.0.15
2.0.15	2.0.1	2.0.15, 2.5.26, 4.0.0	--immutable	Error, manifest & lockfile in conflict

Options

install arguments	description
--ignore-scripts	Do not run any pre~ and post~ scripts. For example preinstall & postinstall.
--install-strategy <STRATEGY>	Define apax install strategy strict (default) or overrideable . Not mention this argument will always fall back to the default or whats defined in the apax.yml.

strict: strict adherence to version requirements / errors if versions do not match

overrideable: more flexible approach to find a compatible version / attempts to satisfy the most requirements, even if not all are strictly met

Examples for apax install

```
devDependencies:
  "@ax/sdk": X.Y.Z #latest major versions 4.1.8 / 3.0.19
```

@ax/sdk": X.Y.Z	installed version
@ax/sdk": 4.0.2	4.0.2
@ax/sdk": ^4.0.2	4.1.8
@ax/sdk": ~4.0.2	4.0.8
@ax/sdk": 3.0.2	3.0.2
@ax/sdk": ^3.0.2	3.0.19
@ax/sdk": ~3.0.2	3.0.19

Update dependencies

command	short	description
apax update		Interactive update of all dependencies according to semantic versioning.

command	short	description
apax update <PACKAGE>		The PACKAGE that should be updated (non-interactive).
apax update --all	-a	Update all package (non interactive) according the semantic versioning
apax update --catalog		Interactive update of all catalogs according to semantic versioning.
apax update --catalog <CATALOG>		The CATALOG that should be updated (non-interactive).

Options

update arguments	short	description
--forceLatest	-f	Add force latest-version for the update ignoring semantic versioning.
--prerelease	-p	Consider including updating to prereleased versions (unstable).

Example for apax update @ax/sdk (non-interactive mode)

```
devDependencies:
  "@ax/sdk": X.Y.Z #latest major versions: 4.1.8 / 3.0.19
```

entry in apax.yml	result
@ax/sdk": 4.0.2	No update
@ax/sdk": ^4.0.2	@ax/sdk ^4.0.2 -> ^4.1.8
@ax/sdk": ~4.0.2	@ax/sdk ~4.0.2 -> ~4.0.3
@ax/sdk": 3.0.2	No update
@ax/sdk": ^3.0.2	@ax/sdk ^3.0.2 -> ^3.0.19
@ax/sdk": ~3.0.2	@ax/sdk ~3.0.2 -> ~3.0.19

Example for apax update @ax/sdk --forceLatest (non-interactive mode)

```
devDependencies:
  "@ax/sdk": x.y.z #latest major versions: 4.1.8 / 3.0.19
```

entry in apax.yml	result
@ax/sdk": 4.0.2	@ax/sdk 4.0.2 -> 4.1.8

entry in apax.yml	result
@ax/sdk": ^4.0.2	@ax/sdk ^4.0.2 -> ^4.1.8
@ax/sdk": ~4.0.2	@ax/sdk ~4.0.2 -> ~4.1.8
@ax/sdk": 3.0.2	@ax/sdk 3.0.2 -> 4.1.8
@ax/sdk": ^3.0.2	@ax/sdk ^3.0.2 -> ^4.1.8
@ax/sdk": ~3.0.2	@ax/sdk ~3.0.2 -> ~4.1.8

Example for apax update <PACKAGE>

```
devDependencies:
  "@ax/sdk": 4.0.2 #latest version 4.0.8
dependencies:
  "@ax/system-timer": 3.0.1 #latest 4.0.1
```

The command `apax update @ax/system-timer -f` will just update the package `@ax/system-timer` from 3.0.1 to version 4.0.1

Add dependencies

command	short	description
apax add <PACKAGE>		The name of the package to add to the projects apax.yml dependencies. It calls implicit an apax install.
apax add <PACKAGE> --dev	-D	Whether to add the package as a devdependencies instead. It call implicitly a apax install.
apax add <CATALOG> --catalog		Add the catalog to the catalogs section. It do NOT call implicitly a apax install --catalog.

Options

add arguments	short	description
--ignore-scripts		Do not run any pre~ and post~ scripts.
--verbose	-v	Show additional information in the terminal.
--tilde	-T	Add a package in a [~] version range, allowing patch updates. Default:[^] version range, allowing minor updates.
--exact	-E	Add a package in an exact version, not allowing updates.

Example for apax add

```
apax add @ax/system@latest --exact
```

Remove dependencies

command	description
<code>apax remove <PACKAGE></code>	The name of the package to remove from the project.
<code>apax remove <CATALOG></code>	The name of the catalog to remove from the project.
<code>apax clean -globally</code>	Remove all temporary files from the project. This includes the .apax folder and contents from the .bin folder which correlate with your apax.yml.

[04] Build

command	description
<code>apax build</code>	Build the project using the ST compiler. Make sure the @ax/sdk or @ax/st package is installed.

Options

build arguments	short	description
<code>--variables=<VARIABLES></code>	<code>-v</code>	Add variables during build, corresponding apax.yml variables will be overridden!
<code>-ignore-scripts</code>		Do not run any pre~ and post~ scripts. For example prebuild & postbuild.

Example for a `postbuild` script

This `postbuild` script executes the unit tests automatically, when the build command was successful.

```
scripts:
  postbuild: apax test
```

[05] Test

command	description
<code>apax test</code>	Run AxUnit tests. Requires the @ax/axunitst package, which is included with e.g. @ax/sdk.

Options

test arguments	short	description
--coverage	-c	Specifies to get coverage.
--ignore-scripts		Do not run any pre~ and post~ scripts. For example pretest & posttest.

[06] Publishing and More

Create a package

command	description
apax pack	Create a package that can be published. Only files specified in the files section of the apax.yml are included.

Options

pack arguments	description
--key=<KEY>	The private<KEY> to sign the package with, instead of taking it from the default file. Begins with "SECRET".
--keyVersion <VERSION>	Optional add a version "v{number}" , default: "v1" .
--ignore-scripts	Do not run any pre~ and post~ scripts. For example prepack & postpack.

apax pack automatically calls apax keygen once, if no <KEY> was provided. A new key with v1 will be generated in the process.

Example apax.yml snippet

```
name: myproject
version: 1.0.0
files:
  - README.md
  - src
```

Given the files section in the apax.yml. Apax pack creates a package myproject-1.0.0.apax.tgz which includes the README.md and the "src"-folder.

Publish a package

command	description
apax publish --package=<PACKAGE> --registry=<URL> --tag <TAG-NAME>	Publish a given package.apax.tgz to the specified package-registry<URL> with an optional tag (default: latest).

Signing of packages

command	description
<code>apax sign --package=<PACKAGE></code>	Sign an existing package. Note that the pack command also signs packages and is preferred; this one is provided for advanced use cases.
<code>apax sign --package=<PACKAGE> --key=<KEY></code>	Same as above. But instead of taking it from the default file, you can use another key

Signed packages protects of manipulated packages. `apax pack` signs every package. If no key exists, a new key will be generated.

Generate keys

command	description
<code>apax keygen</code>	Creates a key-file, consisting of private and public key, if no key-file exists.
<code>apax keygen --override-existing</code>	Override an existing key-file with a new private and public key.

Example of a key pair

```
created: 2023-08-30T16:30:00.000Z
publicKey: 1234effe0123abba5678fefe7890abcd1234def4321dcba00001234affebcda1
privateKey: SECRET9239823897cdfd7d7a1aff4e7e8c8d8a88e88d8765a852b2cc4c7af241618936a6c7e7d58a8b7c7726125c613c3f3e33d3e3a3124ac3c1c1c31fa6a8c7b7a7f2
```

WARNING: never publish the private key!

Deprecate packages

If you no longer wish to maintain a package, or if you would like to encourage users to update to a new or different version, you can deprecate it. Deprecating a package or version will print a message to the terminal when a user installs it anyways. If you want to delete/ unpublish the package instead do it at the package registry directly.

command	description
<code>apax deprecate <PACKAGE> <MESSAGE> --registry <URL></code>	Deprecate a given package.apax.tgz from a specified package-registry<URL>. Use the message field to guide the user.

Optional you can add `--undeprecate` in order to revert the deprecation.

[07] Typical workflows

Create workspace for a PLC application, library or a TIAX workflow via CLI and open it in AxCode

1.	<i>open a command line window</i>	
2.	<code>cd <destination directory></code>	select the destination directory eg. <code>cd \temp\</code>
3.	<code>apax create TEMPLATE NAME</code>	TEMPLATE = [app, lib, tiax], NAME = your workspace name (e.g. <code>myWorkspace</code>)
4.	<code>cd NAME</code>	change to the directory e.g. <code>cd myWorkspace</code>
5.	<code>apax install</code>	install the dependencies
6.	<code>axcode .</code>	open repository in AX Code

Create workspace for a PLC application, library or a TIAX workflow via CLI in an opened AxCode

1.	<i>AxCode is already open</i>	
2.	<i>Select <code>File-->Close</code> folder if a folder is opened</i>	
3.	<i>Open a terminal in AxCode</i>	
4.	<code>cd <destination directory></code>	select the destination directory eg. <code>cd \temp\</code>
5.	<code>apax create <TEMPLATE> <NAME> --here</code>	TEMPLATE = [app, lib, tiax], NAME = your workspace name (e.g. <code>myWorkspace</code>)

please mind that in this case the option `--here` is required for the `apax create` command

[08] Scripting with Apax

In the `apax.yml` you can define a scripting section.

Example `apax.yml` snippet:

```
scripts:
  my-script:
    - apax build
    - apax test
```

To execute the scripts just enter `apax my-script`. In the given example, `apax build` and `apax test` will executed one after each other.

Build-In-Scripts

name	description
------	-------------

name	description
preinstall	executed before <code>apax [install /add / remove / update]</code>
postinstall	executed after <code>apaxx [install /add / remove / update]</code>
prebuild	executed before <code>apax build</code>
postbuild	executed after <code>apax build</code>
prepack	executed before <code>apax pack</code>
postpack	executed after <code>apax pack</code>
postcreate	executed after <code>apax create</code> with the option <code>--postcreate</code> , disabled by default.

Adding `--ignore-scripts` to the corresponding apax commands will disable the execution of these Build-In-Scripts if they are defined.

[09] Contributed Apax Commands

command	short	description
<code>apax --show</code>	<code>-s</code>	Lists all available contributed commands from your repository that are coming from installed packages.
<code>apax <COMMAND-NAME></code>		Execute contributed commands.