# APS Application Certification Criteria

1.2



## Table of Contents

# 1. Introduction

APS Application Certification has been created to distinguish the appropriately packaged application from ones that do not follow the APS packaging guidelines.

APS application packages, which pass APS Application Certification successfully, receive certificate and permission to use special label and logo.

There are two APS Application Certification levels: Silver and Gold. Informally, the difference between these levels is that Gold applications, the highest level, follow the best practices for APS packaging, while Silver applications comply with mandatory rules only. The formal set of requirements for each level is provided below.

# 2. Overview

This section provides the generic outline of the APS Application Certification.

## 2.1. Procedure

The procedure of application package certification requires in-deep package analysis and includes:

- Validation of package consistency
- Validation of certification rules according to this document

Gold-level certification assumes paying of fees.

## 2.2. Result

An outcome of a successful certification is an APS Application Certificate of Compliance. The certificate is a digitally signed XML document that is both machine- and human-readable. Certified application package receives permissions to use one of the labels:

"APS 1.2 Silver" or "APS 1.2 GOLD",

and special APS version-depended logo.

## 2.3. Requirement changes

We continue to work on improving and evolving the APS Application Certification criteria and process, therefore these document is a subject to change. In order to ensure that applications are up-to-date with the APS Application Certification Criteria, every certificate has a limited validity period.

Vendors are encouraged to repeat the certification whenever the requirements are updated. The latest version of this document is always available for download at http://apsstandard.org/.

### 2.4. Application changes

A Certificate of Compliance is bound to a particular APS Package. In the case of a new version or where an update is issued, it should be certified separately.

## 3. Silver certification level

An application package is certified as Silver if it satisfies the following requirements:

### 3.1. Package consistency

An application must be packaged according to the APS format specification. That is, every "MUST" and "SHOULD" requirements in specification text must be satisfied.

### 3.2. Application and vendor information

An application package must contain general information about application and its vendor.

#### 3.2.1. Application homepage

Package metadata must contain URL of the application homepage (`application/homepage` metadata element).

#### 3.2.2. Packager information

Package metadata must contain both URL of the packager homepage and complete packager name. An URL must resolve to a page with content-type of either `text/html` or `application/*-xml`.

#### 3.2.3. Vendor homepage

Package metadata must contain both URL of the vendor homepage and complete vendor name. An URL must resolve to a page with content-type of either `text/html` or `application/*-xml`.

#### 3.2.4. Summary

Package metadata must contain a one-sentence summary of application (`application/presentation/summary` metadata element). An application summary should not contain line feed characters, and should not be longer than 256 UTF-16 characters (counting surrogate pairs for two chars).

#### 3.2.5. Icons

Package metadata must contain an icon for the application (`application/presentation/icon/@path` metadata element must point to the icon in package). The icon should be of one of the following MIME types:

- `image/gif` (not animated)
- `image/jpeg`
- `image/png`

The image dimensions should be 64x64 pixels.

#### 3.2.6. Change log for current version

A package must contain a change log for the current version of application (`application/presentation/changelog` metadata element).

### 3.3. License

A package must contain information about application license.

#### 3.3.1. EULA

A package must contain text of the license agreement for the services aimed to be provided to end users (`application/service/license` metadata element).

#### 3.3.2. EULA Type

A license agreement must be characterized as free or commercial (by using appropriate `free` or `commercial` element).

### 3.4. Application requirements

All resources required by application must be described in metadata file.

#### 3.4.1. Declaring dependencies

If the application needs a resource described in aspect (e.g. PHP or database), it must be declared by supplying aspect-specific requirement in the package metadata.

#### 3.4.2. Requirements for PHP applications

If a package uses PHP aspect, it must declare which versions of PHP are compatible with the application (`php:version` requirement element).

#### 3.4.3. Requirements for ASP.NET applications

If a package users ASP.NET aspect, it must declare which versions of ASP.NET are compatible with the application (`aspnet:version` requirement element).

#### 3.4.4. Requirements for Perl applications

If a package users Perl aspect, it must declare which versions of Perl are compatible with the application (`perl:version` requirement element).

#### 3.4.5. Database requirements

If a package uses database aspect, it must declare default name for each database (`db/default-name` requirement element), and minimum compatible version of database server for each database (`db/server-min-version` metadata attribute).

#### 3.4.6. Web content requirements

If a package declares using of web content, it must declare an approximate size for each part in the web content description (`provision/url-mapping/installed-size` metadata element). The actual content size immediately after service provision must not exceed the content size declared by the application metadata.

### 3.5. Package operability

APS scripts of a package must perform operations they are intended to.

#### 3.5.1. Configuration script

The configuration script must allow installation, configuration and de-installation of the application.

#### 3.5.2. Default installation prefix

Package metadata must contain a default prefix for installation on domain (`url-mapping/default-prefix` metadata element).

#### 3.5.3. Informative errors

Any error message which is returned by configuration or verfication script must help user to recognize, diagnose and recover from errors. Error messages must precisely indicate the problem and suggest a solution.

## 4. Gold certification level

An application package is certified as Gold if it satisfies the requirements of the Silver certification level, and also satisfies the following additional requirements:

### 4.1. Application and vendor information

A package must contain detailed information about application and its vendor.

#### 4.1.1. Service Summary

Every declared service must have a summary information. An application and service summary should not contain line feed characters, and should not be longer than 256 UTF-8 characters (counting surrogate pairs for two chars). If a service represents an application user account, it must be declared in metadata (`service[class="account"]` metadata element).

#### 4.1.2. Description

Package metadata must contain a one-paragraph description of application (`application/presentation/description` metadata element). An application description must not be longer than 1024 UTF-16 characters, and should not be shorter than 256 UTF-16 characters (counting surrogate pairs for two chars).

#### 4.1.3. Categories

Package metadata must contain information about the categories the package belongs to (`application/presentation/categories` metadata element). The categories used must be as defined in the APS Application Categories [http://apsstandard.com/r/doc/aps--application-categories.pdf] document.

#### 4.1.4. Screenshots

A package must contain screenshots for the application (`application/presentation/screenshot` metadata element). Screenshots must be 640 pixels wide.

#### 4.1.5. Full change log

A package must contain change log for all released versions of the package (`application/presentation/changelog` metadata element).

### 4.2. License

A package must contain information about application license.

#### 4.2.1. License management

If the application needs license, it must be declared using license aspect. Every "MUST" and "SHOULD" requirement from the aspect specification must be satisfied.

### 4.3. Localization

A package should be friendly to different languages.

#### 4.3.1. Localization attributes

For each XML element localizable according to the format requirements, there must be a language-neutral entry (i.e. an entry without an `xml:lang` attribute).

#### 4.3.2. Languages

Package metadata must contain information about the languages supported by the application (`application/presentation/languages` metadata element).

### 4.4. Application requirements

All resources required by application must be described in metadata file.

#### 4.4.1. Extended database requirements

If application is capable of sharing database with other applications by using table prefixes, this must be declared in the package by `db/can-use-tables-prefix` requirement element.

#### 4.4.2. Extended web content requirements

If application declares using of web content and it is able to work with any web server, no certain one must be declared in application metadata. Otherwise, either IIS or Apache aspects must be used.

#### 4.4.3. Operating environment

Operating system-dependent applications must require the environment needed (`requirements/env:environment` metadata element).

#### 4.4.4. Virtual container requirements

If a package declares PVC aspect using references to external PVC template source, the template checksum must be provided (`pvc:checksum` metadata element).

### 4.5. Package operability

A package should provide maximum convenience for APS controller to manage.

#### 4.5.1. Extended configuration script

The configuration script must allow to enable and disable application.

#### 4.5.2. Upgrades

If application supports upgrades from previous versions this must be declared in the package by `application/upgrade/@match` element and the appropriate upgrade operation must be implemented in a configure script.

#### 4.5.3. Patches

If application supports updates from previous versions in patch mode, this must be declared in the package by `application/patch/@match` element.

#### 4.5.4. Default settings

Application package should contain enough data to be installed in unattended mode. If package defines a mandatory setting for a service, it must also define default value for the setting (`setting/@default-value` metadata attribute) or declare setting meaning for APS controller (`setting/@class` metadata attribute). Standard values of the attribute should be used in the last case.

#### 4.5.5. Deployment documentation

If provisioning of a service from application package requires a software not specified in requirements (`requirements` element), the package must contain comprehensive deployment instruction for this software and configuration of package. The instruction must also contain the following information:

- Application editions and plans which are implemented in package

- Which APS-compliant panels the package is compatible with (if any restrictions exist)
- Where to get license information (API credentials, keys, etc) if it is required for provisioning
- How to verify provisioning of each service at side of application

The instruction must not contain any license-sensitive information, such as login/password or license key (except trial one).

Root service of application package must declare informational link with `class="deployment-guide"` which must point to the deployment instruction (`presentation/infolinks/link[@class="deployment-guide"]` metadata element).

### 4.6. Application functionality

Application instance itself must be consistent.

#### 4.6.1. Entry-points

It must be possible for user to access all application entry points, including those where login credentials are defined (`entry/variable` metadata element).

#### 4.6.2. Support link

Application must declare valid information link to support service site (`infolinks/link[@class="support"]` metadata element).

#### 4.6.3. Mandatory function tests (web applications)

If application does not have global settings and declares an entry point, it must contain at least one test for functions available after logging in via the entry point. This test must fail if application does not work correcly after logging in and must pass otherwise.

## 5. Possible future extensions

This section contains rules that are not yet part of the certification process. Consider these rules as best practices. Future versions of this document may include these rules in as requirements.

### 5.1. Application and vendor information

A package must contain detailed information about application and its vendor.

#### 5.1.1. Signature

A package must contain content list (`APP-LIST.xml` file), and must be signed by vendor or packager. The certificate must be valid and acknowledged by certification authority.

### 5.2. Package operability

Application package must be the most convenient to manage by etalon APS controller.

#### 5.2.1. Setting semantic

Application package must declare meaning of a setting for APS controller (`setting/@class` metadata attribute). Standard values of the attribute should be used.

#### 5.2.2. Informative script output

Verification script must be declared and its output must be structured (metadata element `verify-script/structured-output`). It must provide error description for controller (`error/system` element), in case no one for user (`error/message`) is provided.

### 5.2.3. Extended informative script output

In case of configuration error caused by the incorrect setting value, verification script output should propose to APS Controller correct value of the one (`output/settings/value` metadata element).

### 5.2.4. Monitoring

Application must provide ability for APS controller to check its heartbeat by declaring resource script (`resource-script` element). The script must return resource with `id="aps-heartbeat"` and `value="0"`, if application instance is able to serve requests only.

### 5.2.5. Backups

Application must declare and provide backup/restore functionality, at level the whole application instance could be restored in case of emergency (`backup-script` element).